迭代二：测试文档

1. 简介

1.1 项目背景

本项目是一个线上众包测试平台，基于基础的众包测试功能提供智能化的增强服务，项目背景详情见需求规格文档。

1.2 测试目的

1. 前端页面渲染
2. 后端数据库交互
3. 后端接口完成度
4. 后端接口正确性
5. 后端接口之间的交互
6. 前端与后端的交互

1.3 测试环境

- springboot版本：2.6.3
- mybatis-plus版本：3.5.1
- mysql版本：8.0.28
- vue版本：2.X
- junit版本：5.8.2
- mock.java版本：1.9.2

1.4 测试范围

- 测试主要为集成测试，后端包括Mapper层单元测试，Service层集成测试，Controller层集成测试。

前端测试包括与后端的交互，即采用postman，人工测试等方式测试前后端交互逻辑。测试内容主要涵盖对迭代二增添或修改接口的测试。

2. 测试计划

2.1 后端测试

1．单元测试：主要对后端的Mapper层进行单元测试，重点测试Mapper层额外编写的sql相关方法逻辑。

2．集成测试：调用后端的Service层和Controller层，以及前端进行层层集成测试。

3. 后端测试

3.1 Mapper层单元测试

3.1.1 测试方向

测试mapper包下的每一个被调用过的方法并调用数据库进行验证。

3.1.2 测试准备

1. 测试方法使用@Transactional注解保证任务原子性，并且在运行完测试用例后将涉及文件操作的文件夹中残留文件删除，这样不容易受外部数据干扰，也不会影响后续的测试进行。
2. 关闭正在运行的后端避免测试途中外部传入新数据影响测试结果。

3.1.3 测试用例

3.1.3.1 TaskMapper

测试编号：0

测试方法：taskMapper.recommendAll

Java 复制代码

```java
    /**
     * 测试推荐全部任务，数据量小的情况下，和selectAll结果相似
     */
    @Test
    @Transactional
    void testRecommendAll() {
        User user = new User("23", "32312", UserIdentity.DELIVER);
        userMapper.insert(user);
        Attribute attribute = new Attribute();
        attribute.setUserId(user.getId());
        attributeMapper.insert(attribute);

        Task task = new Task(user.getId(), "213d", 123, 123, 0, 0L, "dsajdo", "fdejiodfj", "wdioj");
        taskMapper.insert(task);
        assert taskMapper.recommendAll(user.getId()).size() == 1;
        Task task2 = new Task(user.getId(), "213d", 123, 123, 0, 0L, "dsajdo", "fdejiodfj", "fejo");
        taskMapper.insert(task2);
        assert taskMapper.recommendAll(user.getId()).size() == 2;
        Task task3 = new Task(user.getId(), "213d", 123, 123, 0, 0L, "dsajdo", "fdejiodfj", "efjoi");
        taskMapper.insert(task3);
        assert taskMapper.recommendAll(user.getId()).size() == 3;
    }

```

测试编号1：

测试方法：taskMapper.recommendTaskByLabel

Java | 复制代码

```java
    /**
     * 测试根据标签推荐任务，数据量大的情况下，只返回部分推荐数据
     */
    @Test
    @Transactional
    void testRecommendTaskByLabel() {
        User user = new User("23", "32312", UserIdentity.DELIVER);
        userMapper.insert(user);
        Attribute attribute = new Attribute();
        attribute.setUserId(user.getId());
        attributeMapper.insert(attribute);

        long time = System.currentTimeMillis() * 2;
        Task task = new Task(user.getId(), "SMTC", 123, 123, 0, time, "dsajdo", "fdejiodfj", "wo");
        Task task1 = new Task(user.getId(), "MT", 123, 123, 1, time / 4, "dsajdo", "fdejiodfj", "feji");
        taskMapper.insert(task);
        taskMapper.insert(task1);

        Integer tag = null;
        Integer if_finished = null;
        long now_time = System.currentTimeMillis();
        assert taskMapper.recommendTaskByLabel(tag, if_finished, now_time, user.getId()).size() == 2;
        if_finished = 0;
        assert taskMapper.recommendTaskByLabel(tag, if_finished, now_time, user.getId()).size() == 1;
        if_finished = 1;
        assert taskMapper.recommendTaskByLabel(tag, if_finished, now_time, user.getId()).size() == 1;
        tag = 0;
        assert taskMapper.recommendTaskByLabel(tag, if_finished, now_time, user.getId()).size() == 0;
        tag = 1;
        assert taskMapper.recommendTaskByLabel(tag, if_finished, now_time, user.getId()).size() == 1;


        User deliver = new User();
        deliver.setPasswd("passwd");
        deliver.setName("deliver");
        deliver.setEmail("deliver_email");
        deliver.setUserIdentity(UserIdentity.DELIVER);

```

```
        userMapper.insert(deliver);

        Map<String, Object> taskTemplate = new HashMap<>();
        taskTemplate.put("userId", deliver.getId());
        taskTemplate.put("name", "@ctitle(2,6)");
        taskTemplate.put("number", "@integer(50,100)");
        taskTemplate.put("remain", "@integer(0,50)");
        taskTemplate.put("date", System.currentTimeMillis() + 3600 *
1000);
        taskTemplate.put("introduction", "@csentence()");
        taskTemplate.put("tag", tag);
        Mock.set(Task.class, taskTemplate);
        MockObject<Task> mockTask = Mock.get(Task.class);

        for (int i = 0; i < 500; i++) {
            taskMapper.insert(mockTask.getOne());
        }

        List<Task> tasks = taskMapper.recommendTaskByLabel(tag, 0,
System.currentTimeMillis(), user.getId());
        assert tasks.size() == 200;
    }
```

测试编号：2

测试方法：taskMapper.insert

Java 复制代码

```
    /**
     * 测试task插入
     */
    @Test
    void testInsert() {
        User user = new User("23", "32312", UserIdentity.DELIVER);
        userMapper.insert(user);
        Task task = new Task(user.getId(), "213d", 123, 123, 0, 0L,
"dsajdo", "fdejiodfj", "ejwi");
        assert 1 == taskMapper.insert(task);
        taskMapper.deleteByMap(null);
        userMapper.deleteByMap(null);
    }
```

测试编号：3

测试方法：taskMapper.updateById

Java 复制代码

```java
    /**
     * 测试根据ID更新task
     */
    @Test
    @Transactional
    void testUpdateById() {
        System.out.println("wdnmd");
        User user = new User("23", "32312", UserIdentity.DELIVER);
        userMapper.insert(user);
        Task task = new Task(user.getId(), "213d", 123, 123, 0, 0L, "dsajdo", "fdejiodfj", "dhi");
        taskMapper.insert(task);
        int t = task.getId();
        task.setId(234325423);
        assert 0 == taskMapper.updateById(task);
        task.setId(t);
        task.setAurl("67");
        System.out.println(task);
        assert 1 == taskMapper.updateById(task);
        System.out.println(taskMapper.selectById(task.getId()));
        assert "67".equals(taskMapper.selectById(task.getId()).getAurl());
    }
```

测试编号：4

测试方法：taskMapper.selectByMap

Java 复制代码

```java
    /**
     * 根据map查询条件查询task
     */
    @Test
    void testSelectByMap() {
        User user = new User("23", "32312", UserIdentity.DELIVER);
        userMapper.insert(user);
        Task task = new Task(user.getId(), "213d", 123, 123, 0, 0L, "dsajdo", "fdejiodfj", "wdioj");
        taskMapper.insert(task);
        Map<String, Object> select = new HashMap<>();
        select.put("id", task.getId());
        select.put("userId", task.getUserId());
        assert taskMapper.selectByMap(select).size() == 1;
        select.remove("id");
        Task task2 = new Task(user.getId(), "213d", 123, 123, 0, 0L, "dsajdo", "fdejiodfj", "fejo");
        taskMapper.insert(task2);
        assert taskMapper.selectByMap(select).size() == 2;
        Task task3 = new Task(user.getId(), "213d", 123, 123, 0, 0L, "dsajdo", "fdejiodfj", "efjoi");
        taskMapper.insert(task3);
        assert taskMapper.selectByMap(null).size() == 3;
        taskMapper.deleteByMap(null);
        userMapper.deleteByMap(null);
    }
```

测试编号：5

测试方法：taskMapper.selectTaskByLabel

Java | 复制代码

```java
    /**
     * 测试selectTaskByLabel方法，根据标签查询任务
     */
    @Test
    void testSelectTaskByLabel() {
        User user = new User("23", "32312", UserIdentity.DELIVER);
        userMapper.insert(user);
        long time = System.currentTimeMillis() * 2;
        Task task = new Task(user.getId(), "SMTC", 123, 123, 0, time, "dsajdo", "fdejiodfj", "wo");
        Task task1 = new Task(user.getId(), "MT", 123, 123, 1, time / 4, "dsajdo", "fdejiodfj", "feji");
        taskMapper.insert(task);
        taskMapper.insert(task1);
        Integer tag = null;
        Integer if_finished = null;
        long now_time = System.currentTimeMillis();
        assert taskMapper.selectTaskByLabel(tag, if_finished, null, now_time).size() == 2;
        if_finished = 0;
        assert taskMapper.selectTaskByLabel(tag, if_finished, null, now_time).size() == 1;
        if_finished = 1;
        assert taskMapper.selectTaskByLabel(tag, if_finished, null, now_time).size() == 1;
        tag = 0;
        assert taskMapper.selectTaskByLabel(tag, if_finished, null, now_time).size() == 0;
        tag = 1;
        assert taskMapper.selectTaskByLabel(tag, if_finished, null, now_time).size() == 1;
        if_finished = null;
        tag = null;
        String name = "MT";
        assert taskMapper.selectTaskByLabel(tag, if_finished, name, now_time).size() == 2;
        name = "MTC";
        assert taskMapper.selectTaskByLabel(tag, if_finished, name, now_time).size() == 1;
        taskMapper.deleteByMap(null);
        userMapper.deleteByMap(null);
    }
```

3.1.3.2 WorkMapper

测试编号：0

测试方法：workMapper.findUsersTaskPartIn

Java 复制代码

```java
    /**
     * 测试findUsersTaskPartIn方法，查询参与任务x的全部用户
     */
    @Transactional
    @Test
    public void testFindUsersTaskPartIn() {
        final String password = "password", name = "name";
        User deliver = new User("1", password, UserIdentity.DELIVER);
        User user1 = new User("2", password, UserIdentity.WORKER);
        User user2 = new User("3", password, UserIdentity.WORKER);
        User user3 = new User("4", password, UserIdentity.WORKER);
        userMapper.insert(deliver);
        userMapper.insert(user1);
        userMapper.insert(user2);
        userMapper.insert(user3);
        Task task1 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "feji");
        Task task2 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "fejo");
        Task task3 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "fejo");
        Task task4 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "fejo");
        taskMapper.insert(task1);
        taskMapper.insert(task2);
        taskMapper.insert(task3);
        taskMapper.insert(task4);
        // user1 work in task1, task2
        workMapper.insert(new Work(null, user1.getId(), task1.getId(),
WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, user1.getId(), task2.getId(),
WorkStatus.TO_FINISH));
        // user2 work in task1, task2, task3
        workMapper.insert(new Work(null, user2.getId(), task1.getId(),
WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, user2.getId(), task2.getId(),
WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, user2.getId(), task3.getId(),
WorkStatus.TO_FINISH));
        // user3 work in task1, task2, task3, task4
        workMapper.insert(new Work(null, user3.getId(), task1.getId(),
WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, user3.getId(), task2.getId(),
WorkStatus.TO_FINISH));
```

```
34          workMapper.insert(new Work(null, user3.getId(), task3.getId(),
    WorkStatus.TO_FINISH));
35          workMapper.insert(new Work(null, user3.getId(), task4.getId(),
    WorkStatus.TO_FINISH));
36          List<Integer> list = new ArrayList<>();
37          list.add(task1.getId());
38          int cnt = 0;
39          for (UserWorkList workList :
    workMapper.findUsersTaskPartIn(list)) {
40              cnt += workList.getTaskId().size();
41          }
42          assert cnt == (2 + 3 + 4);
43      }
```

测试编号：1

测试方法：workMapper.selectPartTasksByUserId

Java 复制代码

```java
    /**
     * 测试selectPartTasksByUserId，根据用户id查询用户参与的工作
     */
    @Test
    @Transactional
    void testSelectPartTasksByUserId() {
        // 插入角色
        final String password = "password", name = "name";
        User worker = new User("1", password, UserIdentity.WORKER);
        User deliver = new User("2", password, UserIdentity.DELIVER);
        userMapper.insert(worker);
        userMapper.insert(deliver);
        // 插入任务
        Task task1 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "feji");
        Task task2 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "fejo");
        Task task3 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "fejo");
        Task task4 = new Task(deliver.getId(), name, 10, 10, 1, 0L,
"dea", "ejijfw", "fejo");
        taskMapper.insert(task1);
        taskMapper.insert(task2);
        taskMapper.insert(task3);
        taskMapper.insert(task4);
        // 插入工作1
        workMapper.insert(new Work(null, worker.getId(), task1.getId(),
WorkStatus.TO_FINISH));
        // 只有一个工作被查出
        List<Integer> res =
workMapper.selectPartTasksByUserId(worker.getId());
        assert res.size() == 1 && Objects.equals(res.get(0),
task1.getId());
        int[] predictTaskIdList = new int[]{task1.getId(), task2.getId(),
task3.getId(), task4.getId()};
        // 插入工作2，3，4
        workMapper.insert(new Work(null, worker.getId(), task2.getId(),
WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, worker.getId(), task3.getId(),
WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, worker.getId(), task4.getId(),
WorkStatus.TO_FINISH));
        res = workMapper.selectPartTasksByUserId(worker.getId());
        // 4个任务均被查出
        for (int i = 0; i < predictTaskIdList.length; i++) {
```

```
            assert predictTaskIdList[i] == res.get(i);
        }
    }
```

测试编号：2

测试方法：workMapper.insert

Java 复制代码

```java
    /**
     * 测试插入新WORK
     */
    @Test
    public void testInsert() {
        User user0 = new User("666", "jdjwdfnm", UserIdentity.DELIVER);
        User user1 = new User("waoij", "fejiojf", UserIdentity.WORKER);
        User user2 = new User("ooj", "foiwjf", UserIdentity.WORKER);
        userMapper.insert(user0);
        userMapper.insert(user2);
        userMapper.insert(user1);
        Task task = new Task(user0.getId(), "jeiw", 10, 10, 1, 0L, "dea",
"ejijfw", "djo");
        taskMapper.insert(task);
        assert 1 == workMapper.insert(new Work(null, user1.getId(),
task.getId(), WorkStatus.TO_FINISH));
        workMapper.deleteByMap(null);
        taskMapper.deleteByMap(null);
        userMapper.deleteByMap(null);
    }
```

测试编号：3

测试方法：workMapper.selectByMap

Java | 复制代码

```java
    /**
     * 测试根据map查询work
     */
    @Test
    public void testSelectByMap() {
        User user0 = new User("666", "jdjwdfnm", UserIdentity.DELIVER);
        User user1 = new User("waoij", "fejiojf", UserIdentity.WORKER);
        User user2 = new User("ooj", "foiwjf", UserIdentity.WORKER);
        userMapper.insert(user0);
        userMapper.insert(user2);
        userMapper.insert(user1);
        Task task = new Task(user0.getId(), "jeiw", 10, 10, 1, 0L, "dea", "ejijfw", "feji");
        Task task2 = new Task(user0.getId(), "jeiw", 10, 10, 1, 0L, "dea", "ejijfw", "fejo");
        taskMapper.insert(task);
        taskMapper.insert(task2);
        workMapper.insert(new Work(null, user1.getId(), task.getId(), WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, user1.getId(), task2.getId(), WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, user2.getId(), task.getId(), WorkStatus.TO_FINISH));
        workMapper.insert(new Work(null, user2.getId(), task2.getId(), WorkStatus.TO_FINISH));
        Map<String, Object> map = new HashMap<>();
        map.put("userId", user1.getId());
        assert 2 == workMapper.selectByMap(map).size();
        map.put("taskId", task.getId());
        assert 1 == workMapper.selectByMap(map).size();
        workMapper.deleteByMap(null);
        taskMapper.deleteByMap(null);
        userMapper.deleteByMap(null);
    }

```

3.1.3.3 ReportMapper

测试编号：0

测试方法：reportMapper.selectLastByUserIdAndTaskId

Java | 复制代码

```java
    /**
     * 测试selectLastByUserIdAndTaskId，根据用户id和taskId查询用户最近提交的报告
     */
    @Test
    @Transactional
    void testSelectLastByUserIdAndTaskId() {
        final String password = "password", name = "name";
        User worker = new User("1", password, UserIdentity.WORKER);
        User deliver = new User("2", password, UserIdentity.DELIVER);
        userMapper.insert(worker);
        userMapper.insert(deliver);
        // 插入任务
        Task task1 = new Task(deliver.getId(), name, 10, 10, 1, 0L, "dea", "ejijfw", "feji");
        taskMapper.insert(task1);
        // 插入工作
        workMapper.insert(new Work(null, worker.getId(), task1.getId(), WorkStatus.TO_FINISH));
        // 插入报告1
        Report report1 = new Report(worker.getId(), task1.getId(), "67大帝", "czy", "6+");
        reportMapper.insert(report1);
        // 最近的报告是报告1
        assert Objects.equals(reportMapper.selectLastByUserIdAndTaskId(worker.getId(), task1.getId()).getId(), report1.getId());
        // 插入报告2
        Report report2 = new Report(worker.getId(), task1.getId(), "67大帝", "czy", "6+");
        reportMapper.insert(report2);
        // 最近的报告是报告2
        assert Objects.equals(reportMapper.selectLastByUserIdAndTaskId(worker.getId(), task1.getId()).getId(), report2.getId());
    }
```

测试编号：1

测试方法：reportMapper.selectByMap

```java
    /**
     * 测试根据taskId查询task
     */
    @Test
    public void testSelectByTaskId() {
        User user0 = new User("342", "321", UserIdentity.DELIVER);
        User user1 = new User("34", "321", UserIdentity.WORKER);
        User user2 = new User("34333", "321", UserIdentity.WORKER);
        userMapper.insert(user0);
        userMapper.insert(user1);
        userMapper.insert(user2);
        Task task = new Task(user0.getId(), "软工三作业检查", 10, 10, 0, 0L, "a", "b", "dfs");
        taskMapper.insert(task);
        Report report1 = new Report(user1.getId(), task.getId(), "67大帝", "czy", "6+");
        reportMapper.insert(report1);
        Report report2 = new Report(user2.getId(), task.getId(), "67大帝", "czy", "6+");
        reportMapper.insert(report2);
        Map<String, Object> map = new HashMap<>();
        map.put("taskId", task.getId());
        assert reportMapper.selectByMap(map).size() == 2;
        reportMapper.deleteByMap(null);
        taskMapper.deleteById(task.getId());
        userMapper.deleteById(user0.getId());
        userMapper.deleteById(user1.getId());
        userMapper.deleteById(user2.getId());
    }

```

3.1.3.4 CommentMapper

测试编号：0

测试方法：commentMapper.testSelectCommentByReportId

Java 复制代码

```java
    /**
     * 测试根据报告id查询报告下评论
     */
    @Test
    public void testSelectCommentByReportId() {
        // 注册为发包方
        User user = new User(null, "czw@qq.com", "123456", null, UserIdentity.DELIVER, "czw",null,null);
        userMapper.insert(user);
        long time = System.currentTimeMillis() * 2;
        // 发布任务
        Task task = new Task(user.getId(), "SMTC", 123, 123, 0, time, "dsajdo", "fdejiodfj", "wo");
        taskMapper.insert(task);
        // 注册众包工人
        User worker1 = new User(null, "dxy@qq.com", "32312", null, UserIdentity.WORKER, "dxy",null,null);
        userMapper.insert(worker1);
        User worker2 = new User(null, "tzh@qq.com", "32312", null, UserIdentity.WORKER, "tzh",null,null);
        userMapper.insert(worker2);
        User worker3 = new User(null, "hxt@qq.com", "32312", null, UserIdentity.WORKER, "hxt",null,null);
        userMapper.insert(worker3);
        // worker1领取任务
        Work job = new Work(null, worker1.getId(), task.getId(), WorkStatus.TO_FINISH);
        workMapper.insert(job);
        // 在任务下发布报告
        Report report = new Report(null, worker1.getId(), task.getId(), "写得真好啊", "1.2.3.4.", "ios", 0.0, 0);
        reportMapper.insert(report);
        // 在报告下面进行评论
        Comment comment1 = new Comment(null, report.getId(), worker1.getId(), 5.0, "这里有一个问题", null, null);
        commentMapper.insert(comment1);
        Comment comment2 = new Comment(null, report.getId(), worker2.getId(), 4.0, "这里有两个问题", null, null);
        commentMapper.insert(comment2);
        Comment comment3 = new Comment(null, report.getId(), worker3.getId(), 4.9, "这里有三个问题", null, null);
        commentMapper.insert(comment3);
        List<Comment> comments = commentMapper.selectCommentByReportId(report.getId());
        assert comments.get(0).getName().equals("dxy");
```

```
35            assert comments.get(1).getName().equals("tzh");
36            assert comments.get(2).getName().equals("hxt");
37
38            userMapper.deleteByMap(null);
39            taskMapper.deleteByMap(null);
40            workMapper.deleteByMap(null);
41
42            commentMapper.deleteByMap(null);
43            reportMapper.deleteByMap(null);
44        }
```

测试编号：1

测试方法：commentMapper.testSelectComment

Java 复制代码

```java
    /**
     * 测试查询报告下用户发表的comments
     */
    @Test
    @Transactional
    public void testSelectComment() {
        // 注册为发包方
        User user = new User(null, "czw@qq.com", "123456", null, UserIdentity.DELIVER, "czw", null, null);
        userMapper.insert(user);
        long time = System.currentTimeMillis() * 2;
        // 发布任务
        Task task = new Task(user.getId(), "SMTC", 123, 123, 0, time, "dsajdo", "fdejiodfj", "wo");
        taskMapper.insert(task);
        // 注册众包工人
        User worker1 = new User(null, "dxy@qq.com", "32312", null, UserIdentity.WORKER, "dxy", null, null);
        userMapper.insert(worker1);
        User worker2 = new User(null, "tzh@qq.com", "32312", null, UserIdentity.WORKER, "tzh", null, null);
        userMapper.insert(worker2);
        // worker1领取任务
        Work job = new Work(null, worker1.getId(), task.getId(), WorkStatus.TO_FINISH);
        workMapper.insert(job);
        // 在任务下发布报告
        Report report = new Report(null, worker1.getId(), task.getId(), "写得真好啊", "1.2.3.4.", "ios", 0.0, 0);
        reportMapper.insert(report);
        // 在报告下面进行评论
        Comment comment1 = new Comment(null, report.getId(), worker1.getId(), 5.0, "这里有一个问题", null, null);
        commentMapper.insert(comment1);
        Comment comment2 = new Comment(null, report.getId(), worker2.getId(), 4.0, "这里有两个问题", null, null);
        commentMapper.insert(comment2);
        List<Comment> comments = commentMapper.selectComment(report.getId(), worker1.getId());
        assert comments.size() == 1 && comments.get(0).getId().equals(comment1.getId());
        comments = commentMapper.selectComment(report.getId(), worker2.getId());
        assert comments.size() == 1 && comments.get(0).getId().equals(comment2.getId());
```

```
34      }
```

3.1.3.5 UserMapper

测试编号：0

测试方法：userMapper.selectByMap

Java 复制代码

```java
    /**
     * 测试根据email查询用户
     */
    @Test
    void testSelect(){
        Map<String,Object> map=new HashMap<>();
        map.put("email","123");
        assert 0==userMapper.selectByMap(map).size();
    }
```

测试编号：1

测试方法：userMapper.insert

Java 复制代码

```java
    /**
     * 测试插入新用户
     */
    @Test
    void testInsert() {
        User user = new User("123", "123", UserIdentity.DELIVER);
        assert 1 == userMapper.insert(user);
        Map<String, Object> map = new HashMap<>();
        map.put("email", "123");
        userMapper.deleteByMap(map);
    }
```

3.1.3.6 CodeMapper

测试编号：0

测试方法：codeMapper.selectByMap

```java
    /**
     * 测试插入新的code
     */
    @Test
    void testSelect() {
        Map<String, Object> map = new HashMap<>();
        map.put("code", "123");
        assert codeMapper.selectByMap(map).size() == 0;
    }
```

测试编号：1

测试方法：codeMapper.deleteByMap

```java
    /**
     * 测试删除code
     */
    @Test
    void testDelete() {
        codeMapper.insert(new Code("123456"));
        Map<String, Object> map = new HashMap<>();
        map.put("code", "123456");
        assert codeMapper.deleteByMap(map) == 1;
    }
```

3.1.3.7 ReportImageMapper

测试编号：0

测试方法：reportImageMapper.deleteByMap

Java | 复制代码

```java
    /**
     * 测试删除报告下图片
     */
    @Test
    public void testDelete() {
        User user0 = new User("123", "321", UserIdentity.DELIVER);
        User user1 = new User("1", "321", UserIdentity.WORKER);
        userMapper.insert(user0);
        userMapper.insert(user1);
        Task task = new Task(user0.getId(), "软工三作业检查", 10, 10, 0, 0L, "a", "b", "cc");
        taskMapper.insert(task);
        Report report1 = new Report(user1.getId(), task.getId(), "好垃圾啊", "点进去就闪退", "赢麻了的Mate40 pro plus 12+512");
        reportMapper.insert(report1);
        List<String> list = Arrays.asList("123", "3423", "3213");
        for (String s : list) {
            reportImageMapper.insert(new ReportImage(report1.getId(), s));
        }
        Map<String, Object> map = new HashMap<>();
        map.put("reportId", report1.getId());
        assert reportImageMapper.selectByMap(map).size() == 3;
        for (String s : list) {
            map = new HashMap<>();
            map.put("image", s);
            reportImageMapper.deleteByMap(map);
        }
        assert reportImageMapper.selectByMap(map).size() == 0;
        reportImageMapper.deleteByMap(null);
        reportMapper.deleteByMap(null);
        taskMapper.deleteById(task.getId());
        userMapper.deleteById(user0.getId());
        userMapper.deleteById(user1.getId());
    }
```

测试编号：1

测试方法：reportImageMapper.insert

Java 复制代码

```java
    /**
     * 测试为报告插入新的图片
     */
    @Test
    public void testInsert() {
        User user0 = new User("123", "321", UserIdentity.DELIVER);
        User user1 = new User("1", "321", UserIdentity.WORKER);
        userMapper.insert(user0);
        userMapper.insert(user1);
        Task task = new Task(user0.getId(), "软工三作业检查", 10, 10, 0, 0L, "a", "b", "dfs");
        taskMapper.insert(task);
        Report report1 = new Report(user1.getId(), task.getId(), "好垃圾啊", "点进去就闪退", "赢麻了的Mate40 pro plus 12+512");
        reportMapper.insert(report1);
        List<String> list = Arrays.asList("123", "3423", "3213");
        for (String s : list) {
            reportImageMapper.insert(new ReportImage(report1.getId(), s));
        }
        Map<String, Object> map = new HashMap<>();
        map.put("reportId", report1.getId());
        assert reportImageMapper.selectByMap(map).size() == 3;
        reportImageMapper.deleteByMap(null);
        reportMapper.deleteByMap(null);
        taskMapper.deleteById(task.getId());
        userMapper.deleteById(user0.getId());
        userMapper.deleteById(user1.getId());
    }
```

测试编号：2

测试方法：reportImageMapper.selectByMap

```java
    /**
     * 测试根据map查询报告图片
     */
    @Test
    public void testSelectByMap() {
        User user0 = new User("342", "321", UserIdentity.DELIVER);
        User user1 = new User("34", "321", UserIdentity.WORKER);
        userMapper.insert(user0);
        userMapper.insert(user1);
        Task task = new Task(user0.getId(), "软工三作业检查", 10, 10, 0, 0L, "a", "b", "ddji");
        taskMapper.insert(task);
        Report report1 = new Report(user1.getId(), task.getId(), "67大帝", "czy", "6+");
        reportMapper.insert(report1);
        List<String> list = Arrays.asList("123", "3423", "3213");
        for (String s : list) {
            reportImageMapper.insert(new ReportImage(report1.getId(), s));
        }
        Map<String, Object> map = new HashMap<>();
        map.put("reportId", report1.getId());
        assert reportImageMapper.selectByMap(map).size() == 3;
        reportImageMapper.deleteByMap(null);
        reportMapper.deleteByMap(null);
        taskMapper.deleteById(task.getId());
        userMapper.deleteById(user0.getId());
        userMapper.deleteById(user1.getId());
    }
```

3.2 Service层集成测试

3.2.1 测试方向

通过调用方法以及assert断言测试Service层中各方法的准确性。

3.2.2 测试准备

1. 测试方法使用@Transactional注解保证任务原子性，并且在运行完测试用例后将涉及文件操作的文件夹中残留文件删除，保证下一次测试不受影响。
2. 关闭正在运行的后端避免测试途中外部传入新数据影响测试结果。

3.2.3 测试用例

3.2.3.1 WorkService

测试编号：0

测试方法：workService.testTaskFinish

Java 复制代码

```java
    /**
     * 测试定时任务，不能加@Transaction注解，已手动删除数据
     */
    @Test
    public void testTaskFinish() {
        // 创建用户，发包方，工人
        User deliver = new User();
        deliver.setPasswd("passwd");
        deliver.setName("deliver");
        deliver.setEmail("deliver_email");
        deliver.setUserIdentity(UserIdentity.DELIVER);

        User worker = new User();
        worker.setPasswd("passwd");
        worker.setName("worker");
        worker.setEmail("worker_email");
        worker.setUserIdentity(UserIdentity.DELIVER);

        userMapper.insert(deliver);
        userMapper.insert(worker);

        // 为工人创建属性条目
        Attribute attribute = new Attribute();
        attribute.setUserId(worker.getId());
        attributeMapper.insert(attribute);

        // 创建任务1，任务2，任务1在3s后过期，任务2在8s后过期
        TaskDTO taskOverTime = new TaskDTO();
        taskOverTime.setUserId(deliver.getId());
        taskOverTime.setName("taskOverTime");
        taskOverTime.setNumber(100);
        taskOverTime.setTag(1);
        taskOverTime.setDevice(DeviceType.HarmonyOS);
        taskOverTime.setIntroduction("introduction");
        taskOverTime.setDate(System.currentTimeMillis() + 1000 * 3);

        TaskDTO taskInTime = new TaskDTO();
        taskInTime.setUserId(deliver.getId());
        taskInTime.setName("taskInTime");
        taskInTime.setNumber(100);
        taskInTime.setTag(1);
        taskInTime.setDevice(DeviceType.HarmonyOS);
        taskInTime.setIntroduction("introduction");
        taskInTime.setDate(System.currentTimeMillis() + 1000 * 8);

```

```
46          int taskOverTimeId = ((TaskVO)
    taskService.issueTask(taskOverTime).getData().get(0)).getId();
47          int taskInTimeId = ((TaskVO)
    taskService.issueTask(taskInTime).getData().get(0)).getId();
48
49          // 工人领取任务1，任务2
50          WorkDTO partTaskOverTime = new WorkDTO();
51          partTaskOverTime.setTaskId(taskOverTimeId);
52          partTaskOverTime.setUserId(worker.getId());
53          workService.partTask(partTaskOverTime);
54
55          WorkDTO partTaskInTime = new WorkDTO();
56          partTaskInTime.setTaskId(taskInTimeId);
57          partTaskInTime.setUserId(worker.getId());
58          workService.partTask(partTaskInTime);
59
60          // 5s后，任务1过期
61          try {
62              Thread.sleep(1000 * 5);
63          } catch (InterruptedException e) {
64              e.printStackTrace();
65          }
66
67          // 工作1失败，工作2仍待完成
68          assert workService.findWorkStatus(worker.getId(),
    taskOverTimeId).equals(WorkStatus.FAIL);
69          assert workService.findWorkStatus(worker.getId(),
    taskInTimeId).equals(WorkStatus.TO_FINISH);
70
71          // 工人为工作2提交报告
72          ReportDTO reportDTO = new ReportDTO();
73          reportDTO.setUserId(worker.getId());
74          reportDTO.setTaskId(taskInTimeId);
75          reportDTO.setSteps("steps");
76          reportDTO.setNote("note");
77          reportService.commitReport(reportDTO);
78
79          // 5s后，任务2结束，工作2完成
80          try {
81              Thread.sleep(1000 * 5);
82          } catch (InterruptedException e) {
83              e.printStackTrace();
84          }
85
86          // 两个工作结束，工人属性增加2
87          Map<String, Object> selectByUserId = new HashMap<>();
88          selectByUserId.put("userId", worker.getId());
```

```java
        assert workService.findWorkStatus(worker.getId(), taskInTimeId).equals(WorkStatus.FINISH);
        assert attributeMapper.selectByMap(selectByUserId).get(0).getHarmonyos() == 2;

        // 删除用户
        userMapper.deleteById(worker.getId());
        userMapper.deleteById(deliver.getId());

        // 删除任务
        taskMapper.deleteById(taskInTimeId);
        taskMapper.deleteById(taskOverTimeId);

        // 删除工作
        Map<String, Object> deleteByUserId = new HashMap<>();
        deleteByUserId.put("userId", worker.getId());
        workMapper.deleteByMap(deleteByUserId);

        // 删除报告
        reportMapper.deleteByMap(deleteByUserId);

        //删除属性
        attributeMapper.deleteByMap(deleteByUserId);

    }
```

测试编号：1

测试方法：workService.findTaskByUserId

```
Java | 复制代码
```

测试编号：2

测试方法：workService.partTask

```
Java | 复制代码
```

3.2.3.2 TaskService

测试编号：0

测试方法：taskService.testRecommendAll

```
Java | 复制代码
```

测试编号：1
测试方法：taskService.updateTaskInfo

Java 复制代码

测试编号：2
测试方法：taskService.issueTask

Java 复制代码

测试编号：3
测试方法：taskService.findTaskByTaskIdList

Java 复制代码

测试编号：4
测试方法：taskService.findAllTasks

Java 复制代码

测试编号：5
测试方法：taskService.selectTaskByLabel

Java 复制代码

3.2.3.3 ReportService

测试编号：0
测试方法：reportService.testGetCommentsUnderReport

Java 复制代码

测试编号：1
测试方法：reportService.testGiveMarkAndComment

Java 复制代码

测试编号：2
测试方法：reportService.testFindSimilarReports

Java 复制代码

测试编号：3
测试方法：reportService.testGiveAnnotation

Java 复制代码

测试编号：4
测试方法：reportService.testShowAnnotation

Java 复制代码

测试编号：5
测试方法：reportService.testFindCoworkers

Java 复制代码

测试编号：6
测试方法：reportService.commitReport，reportService.updateReport

Java 复制代码

测试编号：7
测试方法：reportService.lookReports

Java 复制代码

测试编号：8
测试方法：reportService.lookReports

Java 复制代码

3.2.3.4 AttributeService

测试编号：0
测试方法：attributeService.testInit

Java 复制代码

3.2.3.5 UserService

测试编号：0

测试方法：userService.register

Java 复制代码

测试编号：1

测试方法：userService.login

Java 复制代码

测试编号：2

测试方法：userService.codeJudge

Java 复制代码

测试编号：3

测试方法：userService.getIdentityByUserId

Java 复制代码

3.3 Controller层集成测试

3.3.1 测试方向

通过调用方法以及assert断言测试controller层中各方法的准确性。

3.3.2 测试准备

1. 测试方法使用@Transactional注解保证任务原子性，并且在运行完测试用例后将涉及文件操作的文件夹中残留文件删除，保证下一次测试不受影响。
2. 关闭正在运行的后端避免测试途中外部传入新数据影响测试结果。

3.3.3 测试用例

3.3.3.1 UserController

测试编号：0

测试方法：userController.testInit

Java 复制代码

测试编号：1
测试方法：userController.register

Java 复制代码

测试编号：2
测试方法：userController.register

Java 复制代码

测试编号：3
测试方法：userController.login

Java 复制代码

3.3.3.2 ReportController

测试编号：0
测试方法：reportController.testFindSimilarReportsFromSameTask

Java 复制代码

测试编号：1
测试方法：reportController.testGiveMarkAndComment

Java 复制代码

测试编号：2
测试方法：reportController.testGetCommentsUnderReport

Java 复制代码

测试编号：3
测试方法：reportController.testFindLowQualityReportsFromSameTask

Java 复制代码

测试编号：4
测试方法：reportController.commitReport

Java 复制代码

测试编号：5
测试方法：reportController.lookReports

Java 复制代码

3.3.3.3 TaskController

测试编号：0
测试方法：taskController.testRecommendTasksByLabel

Java 复制代码

测试编号：1
测试方法：taskController.issueTask、taskController.searchPublishedTask

Java 复制代码

测试编号：2
测试方法：taskController.searchTaskForWorker

Java 复制代码

测试编号：3
测试方法：taskController.searchTaskForWorker

Java 复制代码

测试编号：4
测试方法：taskController.updateTaskInfo

Java 复制代码

测试编号：5
测试方法：taskController.issueTask

Java 复制代码

测试编号：6

测试方法：taskController.findAllTasks

Java 复制代码

测试编号：7

测试方法：taskController.selectTaskByLabel

Java 复制代码

3.3.3.4 WorkController

测试编号：0

测试方法：workController.partTask

Java 复制代码

4.测试结果

Sessionsbackend-crowdsourcedtesting

backend-crowdsourcedtesting

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.collect.vo	1,031	613	37%	80	2	2%	151	220	104	201	110	179	1	10
com.collect.dto	790	824	51%	71	8	10%	90	196	70	188	51	153	1	7
com.collect.util	569	643	53%	62	50	45%	67	108	135	257	21	52	1	8
com.collect.po	496	923	65%	22		0%	34	186	32	186	23	175	1	11
com.collect.service.impl	490	1,905	80%	85	104	55%	79	152	109	464	13	54	0	9
com.collect.config	417	249	37%	75	3	4%	55	79	1	39	16	40	0	4
com.collect.vo.http	387	231	37%			n/a	30	42	48	82	30	42	4	7
com.collect.aop	214		4%	30		0%	18	21	42	45	3	6	0	2
com.collect.mapper	180	971	84%	33	63	66%	28	91	6	174	9	43	0	4
com.collect.po.enums	144	312	68%	10		0%	18	34	15	45	11	27	0	5
com.collect.controller	115	176	60%	11	13	54%	22	46	23	58	13	34	0	4
com.collect.util.reportStrategy	46	208	82%	6	14	70%	6	15	7	48	0	5	0	3
com.collect.util.strategy	29	41	59%	2	2	50%	2	6	4	10	1	4	0	2
com.collect.exception			0%			n/a	3	3	2	2	3	3	1	1
com.collect.annotation	43		90%			n/a	1	4	0	2	1	4	0	1
com.collect			38%			n/a	1	2	2	3	1	2	0	1
Total	4,932 of 12,083		59%	487 of 746		35%	605	1,205	600	1,804	306	823	9	79

Created with JaCoCo 0.7.7.201606060606

测试覆盖率在59%，考虑到我们使用了mybatis-plus框架，会有一些自动生成的类和方法没有测试到，比mybatis框架和JPA有更低的测试覆盖率也很正常。
另外，对于迭代一使用了、而迭代二没使用的代码，我们并没有删除，而是遵循了开闭原则，仅仅对方法加上@Deprecated注解，这些方法在测试时也是不在测试范围内的，从而导致了测试覆盖率较低。