1. 修正答题bug

> *在选完答案，用户提交的时候，如果用户连续点击按钮，就会造成多次提交，那么我们的答题进度就会遭到破坏，所以需要判断如果用户已经答过题了，那么就不能继续执行业务了。*
>
> *这个bug告诉我们一个道理，永远不要相信前端，永远不要相信用户。*

```
 //修正bug 当用户连续点击submit提交时，我们应该判断 如果此题已经回答，则不能走以下的逻辑
        UserPracticeDomain userPracticeDomain = this.topicDomainRepository.createUserPracticeDomain(null);
        UserPractice practice = userPracticeDomain.findUserPracticeByTopicId(userId, topicId, practiceId);
        if (practice.getPStatus() != 0){
            return CallResult.fail(-999,"此题已经回答过了");
        }
```

> *进度未进行更新*

```
 if (userHistory.getProgress() >= userHistory.getTopicTotal()){
            //已经是最后一道题了,更新状态已完成和完成时间
            historyDomain.updateUserHistoryStatus(userHistory.getId(), HistoryStatus.FINISHED.getCode(), System.currentTimeMillis());
            finish = true;
        }else{
            //这 需要进行进度更新

 historyDomain.updateUserHistoryProgress(userHistory.getId());
        }
```

2. 我的课程

> *查询当前用户已购买的课程*

2.1 Api

请求路径: /api/course/myCourse

参数: 无

返回值:


```
{
    "code":2000000000,
    "message":"default success",
    "result":[
        {
            "courseName":"七年级道法上",
            "status":1,
            "buyTime":"2021年11月23日",
            "expireTime":"2022年11月23日",
            "studyCount":0,
            "courseId":6
        },
        {
            "courseName":"七年级道法下",
            "status":1,
            "buyTime":"2021年11月23日",
            "expireTime":"2022年11月23日",
            "studyCount":0,
            "courseId":7
        }
    ],
    "success":true
}
```


```
package com.xiaopizhu.tiku.model;

import lombok.Data;

@Data
```

```
public class UserCourseModel {
    private String courseName;
    /**
     * 1 有效
     * 2，已过期
     */
    private Integer status;

    private String buyTime;

    private String expireTime;
    private Integer studyCount;
    private Long courseId;
}
```

2.2 编码

2.2.1 Controller

```
@PostMapping(value = "myCourse")
    public CallResult myCourse(){
        CourseParam courseParam = new CourseParam();
        return courseService.myCourse(courseParam);
    }
```

2.2.2 Service

```
@Override
    public CallResult myCourse(CourseParam courseParam) {
        CourseDomain courseDomain =
courseDomainRepository.createDomain(courseParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return courseDomain.myCourse();
            }
        });
    }
```

2.2.3 Domain

```
public CallResult<Object> myCourse() {
        Long userId = UserThreadLocal.get();
        List<UserCourse> userCourseList =
this.courseDomainRepository.createUserCourseDomain(new
UserCourseParam()).findCourseByUserId(userId);
        List<UserCourseModel> userCourseModelList = new ArrayList<>();
        Long currentTime = System.currentTimeMillis();
        for (UserCourse order : userCourseList){
            Long courseId = order.getCourseId();
            Course course =
this.courseDomainRepository.findCourseById(courseId);
            UserCourseModel userCourseModel = new UserCourseModel();
            userCourseModel.setCourseName(course.getCourseName());
            if (currentTime > order.getExpireTime()){
                userCourseModel.setStatus(2);
            }else {
                userCourseModel.setStatus(1);
            }
            userCourseModel.setBuyTime(new
DateTime(order.getCreateTime()).toString("yyyy年MM月dd日"));
            userCourseModel.setExpireTime(new
DateTime(order.getExpireTime()).toString("yyyy年MM月dd日"));
            List<SubjectModel> subjectInfoByCourseId =
this.courseDomainRepository.createSubjectDomain(new
SubjectParam()).findSubjectModelListByCourseId(courseId);
            Integer count =
this.courseDomainRepository.createHistoryDomain(null).countUserHistoryBy
SubjectList(userId,subjectInfoByCourseId);
            userCourseModel.setStudyCount(count);
            userCourseModel.setCourseId(courseId);
            userCourseModelList.add(userCourseModel);
        }
        return CallResult.success(userCourseModelList);
    }
```

```
 public UserCourseDomain createUserCourseDomain(UserCourseParam
userCourseParam) {
        return userCourseDomainRepository.createDomain(userCourseParam);
    }
```

UserCourseDomain:

```
public List<UserCourse> findCourseByUserId(Long userId) {
        return userCourseDomainRepository.findCourseByUserId(userId);
    }
```

```
public List<UserCourse> findCourseByUserId(Long userId) {
        LambdaQueryWrapper<UserCourse> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserCourse::getUserId,userId);
        return userCourseMapper.selectList(queryWrapper);
    }
```

UserHistoryDomain:

```
public Integer countUserHistoryBySubjectList(Long userId,
List<SubjectModel> subjectModelList) {

        return
userHistoryDomainRepository.countUserHistoryBySubjectList(userId,subject
ModelList);
    }
```

```
public Integer countUserHistoryBySubjectList(Long userId,
List<SubjectModel> subjectModelList) {
        LambdaQueryWrapper<UserHistory> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserHistory::getUserId,userId);
        List<Long> subjectList =
subjectModelList.stream().map(SubjectModel::getId).collect(Collectors.to
List());
        queryWrapper.in(UserHistory::getSubjectId,subjectList);
        return userHistoryMapper.selectCount(queryWrapper);
    }
```

2.3 测试

3. 我的学习

3.1 Api

请求路径：/api/topic/practiceHistory

参数：{page: 1, pageSize: 10}

返回数据:


```
{
    "code":2000000000,
    "message":"default success",
    "result":{
        "pageCount":1,
        "page":1,
        "total":1,
        "list":[
            {
                "id":1465353052000100354,
                "createTime":"2021-11-30 00:12:47",
                "subjectUnitList":[
                    1,
                    2,
                    3,
                    4,
                    5,
                    6,
                    7,
                    8
                ],
                "subjectName":"道法 七年级 下",
                "subjectId":8,
                "finishTime":"",
                "useTime":"",
                "historyStatus":1,
                "status":0
            }
        ]
    },
    "success":true
}
```

```java
package com.mszlu.xt.web.model;

import lombok.Data;

import java.util.List;

@Data
public class UserHistoryModel {
    private Long id;
    private String createTime;
    private List<Integer> subjectUnitList;
    private String subjectName;
    private Long subjectId;
    private String finishTime;
    private String useTime;
    //1 未完成 2 已完成 3 已取消
    private Integer historyStatus;
    //0 正常 1 已过期
    private Integer status;
}
```

3.2 编码

3.2.1 Controller

```java
  @PostMapping(value = "practiceHistory")
    public CallResult practiceHistory(@RequestBody TopicParam
topicParam){
        return topicService.practiceHistory(topicParam);
    }
```

3.2.2 Service

```
@Override
    public CallResult practiceHistory(TopicParam topicParam) {
        TopicDomain topicDomain =
this.topicDomainRepository.createDomain(topicParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {

            @Override
            public CallResult<Object> doAction() {
                return topicDomain.practiceHistory();
            }
        });
    }
```

3.2.3 Domain

```
 public CallResult<Object> practiceHistory() {
        Long userId = UserThreadLocal.get();
        Integer page = this.topicParam.getPage();
        Integer pageSize = this.topicParam.getPageSize();
        Page<UserHistory> userHistoryPage =
this.topicDomainRepository.createUserHistoryDomain(null).findUserHistory
List(userId,page,pageSize);

        List<UserHistory> userHistoryList =
userHistoryPage.getRecords();

        List<UserHistoryModel> userHistoryModelList = new ArrayList<>();

        for (UserHistory userHistory : userHistoryList) {
            UserHistoryModel userHistoryModel = new UserHistoryModel();
            userHistoryModel.setId(userHistory.getId());
            userHistoryModel.setCreateTime(new
DateTime(userHistory.getCreateTime()).toString("yyyy-MM-dd HH:mm:ss"));

 userHistoryModel.setHistoryStatus(userHistory.getHistoryStatus());
            SubjectModel subject =
this.topicDomainRepository.createSubjectDomain(null).findSubject(userHis
tory.getSubjectId());
            userHistoryModel.setSubjectName(subject.getSubjectName()+"
"+subject.getSubjectGrade()+" "+subject.getSubjectTerm());
            userHistoryModel.setSubjectId(subject.getId());
```

```
            userHistoryModel.setFinishTime(userHistory.getFinishTime()
== 0 ? "":new DateTime(userHistory.getFinishTime()).toString("yyyy-MM-dd
HH:mm:ss"));
            List<Integer> subjectUnitList =
JSON.parseArray(userHistory.getSubjectUnits(), Integer.class);
            userHistoryModel.setSubjectUnitList(subjectUnitList);
            userHistoryModel.setUseTime(userHistory.getFinishTime() == 0
? "":useTime(userHistory.getFinishTime(),userHistory.getCreateTime()));
            List<Long> courseIdList =
this.topicDomainRepository.createCourseDomain(null).findCourseIdListBySu
bjectId(subject.getId());
            int count =
this.topicDomainRepository.createUserCourseDomain(null).countUserCourseI
nCourseIdList(userId,courseIdList,System.currentTimeMillis());
            if (count > 0){
                //判断是否此学习 还在购买的 有效期内
                userHistoryModel.setStatus(0);
            }else{
                userHistoryModel.setStatus(1);
            }
            userHistoryModelList.add(userHistoryModel);
        }

        ListPageModel<UserHistoryModel> listModel = new ListPageModel<>
();
        listModel.setList(userHistoryModelList);
        listModel.setSize(userHistoryPage.getTotal());
        listModel.setPage(page);
        listModel.setPageSize(pageSize);
        listModel.setPageCount(userHistoryPage.getPages());
        return CallResult.success(listModel);
    }
```

UserHistoryDomain:

```
 public Page<UserHistory> findUserHistoryByUserId(Long userId, Integer
page, Integer pageSize) {

        return
userHistoryDomainRepository.findUserHistoryByUserId(userId,page,pageSize
);
    }
```

```
 public Page<UserHistory> findUserHistoryByUserId(Long userId, Integer
page, Integer pageSize) {
        LambdaQueryWrapper<UserHistory> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserHistory::getUserId,userId);
        return userHistoryMapper.selectPage(new Page<>(page,pageSize),
queryWrapper);
    }
```

3.3 测试

4. 错题本

> *当做错题后，会将其加入错题本*

4.1 修正加入错题本bug

1. 首先将shardingjdbc更新为新版本

```
<shardingsphere.version>5.0.0</shardingsphere.version>
```

2. 加入错题本代码，少了subjectId的设值

```
//加入错题本
            UserProblem userProblem =
userProblemDomain.getUserProblem(userId,topicId);
            if (userProblem == null){
                userProblem = new UserProblem();
                userProblem.setErrorCount(1);

 userProblem.setErrorStatus(ErrorStatus.NO_SOLVE.getCode());
                userProblem.setUserId(userId);
                userProblem.setTopicId(topicId);

 userProblem.setSubjectId(topicParam.getSubjectId());
                userProblem.setErrorAnswer(answer);
                //这少了
```

```
                userProblem.setSubjectId(topic.getTopicSubject());

 userProblem.setErrorTime(System.currentTimeMillis());
                int insertCount =
userProblemDomain.saveUserProblem(userProblem);
```

3. 更新计数，使用方法错误

```
//更新错题数量
public void updateUserProblemErrorCount(Long userId, Long topicId,
String answer) {
        UpdateWrapper<UserProblem> update = Wrappers.update();
        update.eq("user_id",userId);
        update.eq("topic_id",topicId);
        update.setSql(true,"error_count=error_count+1");
        update.set("error_answer",answer);
        userProblemMapper.update(null, update);
    }
```

```
//更新错题数量
public void updateUserHistoryErrorCount(Long userHistoryId) {
        UpdateWrapper<UserHistory> update = Wrappers.update();
        update.eq("id",userHistoryId);
        update.setSql(true,"error_count=error_count+1");
        this.userHistoryMapper.update(null, update);
    }
```

```
//更新进度
public void updateUserHistoryProgress(Long historyId) {
        UpdateWrapper<UserHistory> update = Wrappers.update();
        update.eq("id",historyId);
        update.setSql(true,"progress=progress+1");
        this.userHistoryMapper.update(null, update);
    }
```

4. 测试

4.2 Api

请求路径: /api/topic/userProblemSearch

参数：{page: 1, pageSize: 10, subjectTerm: "全部", subjectGrade: "全部", subjectName: "全部"}

返回数据:


```
{
    "code":2000000000,
    "message":"default success",
    "result":{
        "pageCount":10,
        "page":1,
        "pageSize":0,
        "size":3,
        "list":[
            {
                "problemId":1465369809028952066,
                "topic":{
                    "id":"1438886805117128706",
                    "topicTitle":"漫画《我从这里腾飞》启示我们要（    ）",
                    "topicType":1,
                    "topicImgList":[
                        "http://static.lzxtedu.com/7a2/1157.png"
                    ],
                    "topicStar":1,
                    "topicAreaPro":"",
                    "topicAreaCity":"嘉陵区",
                    "fillBlankAnswer":null,
                    "fillBlankTopicChoice":0,
                    "radioChoice":[
                        {
                            "A":{
                                "content":"增强合作意识，学会合作学习",
                                "imageList":[

                                ]
                            }
                        },
                        {
                            "B":{
                                "content":"克服考试过度焦虑，轻装上阵",
                                "imageList":[

                                ]
                            }
                        },
                        {
                            "C":{
```

```
                        "content":"学会学习，培养良好学习习惯",
                        "imageList":[

                        ]
                    }
                },
                {
                    "D":{
                        "content":"激发探究兴趣，加强自主学习",
                        "imageList":[

                        ]
                    }
                }
            ],
            "mulChoice":null,
            "answer":null,
            "analyze":null,
            "subject":null,
            "lastUpdateTime":"2021-10-03",
            "userAnswer":null,
            "pstatus":null
        },
        "subject":{
            "id":8,
            "subjectName":"道法",
            "subjectGrade":"七年级",
            "subjectTerm":"下",
            "subjectUnitList":null
        },
        "errorCount":3,
        "errorAnswer":"B",
        "errorTime":"2021-11-30 01:19:22"
    },
    {
        "problemId":1465374066159411201,
        "topic":{
            "id":"14388868489979371555",
            "topicTitle":"2019年9月17日，国家主席习近平签署主席令，根据主席令，授予于敏、申纪兰（女）、孙家栋、李延年、张富清、袁隆平、黄旭华、屠呦呦（女）“共和国勋章”。经中共中央批准，中华人民共和国国家勋章和国家荣誉称号颁授仪式于2019年9月29日上午10时在人民大会堂隆重举行。褒扬这些为党和人民事业作出贡献的杰出人士的代表，旨在（　　）\n①用鲜活的事例讲好中国故事，传播好中国声音\n②阐发以改革创新为核心的民族精神，构筑中国价值\n③传承中华文化基因，发展中国特色社会主义文化\n④弘扬伟大创造精神、奋斗精神、团结精神和梦想精神",
```

```
"topicType":1,
"topicImgList":[

],
"topicStar":1,
"topicAreaPro":"",
"topicAreaCity":"天河区",
"fillBlankAnswer":null,
"fillBlankTopicChoice":0,
"radioChoice":[
    {
        "A":{
            "content":"①②③",
            "imageList":[

            ]
        }
    },
    {
        "B":{
            "content":"②③④",
            "imageList":[

            ]
        }
    },
    {
        "C":{
            "content":"①③④",
            "imageList":[

            ]
        }
    },
    {
        "D":{
            "content":"①②④",
            "imageList":[

            ]
        }
    }
],
"mulChoice":null,
"answer":null,
```

```
                    "analyze":null,
                    "subject":null,
                    "lastUpdateTime":"2021-10-03",
                    "userAnswer":null,
                    "pstatus":null
                },
                "subject":{
                    "id":8,
                    "subjectName":"道法",
                    "subjectGrade":"七年级",
                    "subjectTerm":"下",
                    "subjectUnitList":null
                },
                "errorCount":1,
                "errorAnswer":"D",
                "errorTime":"2021-11-30 01:36:17"
            },
            {
                "problemId":1465374093661462529,
                "topic":{
                    "id":"1438886821755932675",
                    "topicTitle":"下列不属于扩大自己的朋友圈的好方式的是
（    ）",
                    "topicType":1,
                    "topicImgList":[

                    ],
                    "topicStar":1,
                    "topicAreaPro":"",
                    "topicAreaCity":"从化区",
                    "fillBlankAnswer":null,
                    "fillBlankTopicChoice":0,
                    "radioChoice":[
                        {
                            "A":{
                                "content":"参加公益活动",
                                "imageList":[

                                ]
                            }
                        },
                        {
                            "B":{
                                "content":"利用课余时间，帮助后进的同学补习功
课",
```

```
                                "imageList":[

                                ]
                            }
                        },
                        {
                            "C":{
                                "content":"上网玩游戏",
                                "imageList":[

                                ]
                            }
                        },
                        {
                            "D":{
                                "content":"积极参加班级的各项活动",
                                "imageList":[

                                ]
                            }
                        }
                    ],
                    "mulChoice":null,
                    "answer":null,
                    "analyze":null,
                    "subject":null,
                    "lastUpdateTime":"2021-10-03",
                    "userAnswer":null,
                    "pstatus":null
                },
                "subject":{
                    "id":8,
                    "subjectName":"道法",
                    "subjectGrade":"七年级",
                    "subjectTerm":"下",
                    "subjectUnitList":null
                },
                "errorCount":1,
                "errorAnswer":"D",
                "errorTime":"2021-11-30 01:36:23"
            }
        ]
    },
    "success":true
}
```

```java
package com.mszlu.xt.web.model;

import lombok.Data;

@Data
public class UserProblemModel {
    private Long problemId;
    private TopicModelView topic;
    private SubjectModel subject;
    private Integer errorCount;
    private String errorAnswer;
    private String errorTime;
}
```

4.3 编码

4.3.1 Controller

```java
@PostMapping(value = "userProblemSearch")
    public CallResult userProblemSearch(@RequestBody TopicParam
topicParam){
        return topicService.userProblemSearch(topicParam);
    }
```

4.3.2 Service

```
@Override
    public CallResult userProblemSearch(TopicParam topicParam) {
        TopicDomain topicDomain =
this.topicDomainRepository.createDomain(topicParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {

            @Override
            public CallResult<Object> doAction() {
                return topicDomain.userProblemSearch();
            }
        });
    }
```

4.3.3 Domain

```
public CallResult<Object> userProblemSearch() {
        int page = this.topicParam.getPage();
        int pageSize = this.topicParam.getPageSize();
        String subjectName = this.topicParam.getSubjectName();
        String subjectGrade = this.topicParam.getSubjectGrade();
        String subjectTerm = this.topicParam.getSubjectTerm();
        Long userId = UserThreadLocal.get();
        Long searchSubjectId =
this.topicDomainRepository.createSubjectDomain(null).findSubjectByInfo(s
ubjectName,subjectGrade,subjectTerm);
        if (searchSubjectId == null){
            Page<UserProblem> userProblemListPage =
this.topicDomainRepository.createUserProblem(null).findUserProblemList(u
serId, ErrorStatus.NO_SOLVE.getCode(),page,pageSize);
            List<UserProblemModel> userProblemModelList = new
ArrayList<>();
            List<UserProblem> userProblemList =
userProblemListPage.getRecords();
            for (UserProblem userProblem : userProblemList){
                Long topicId = userProblem.getTopicId();
                Long subjectId = userProblem.getSubjectId();
                Topic topic =
this.topicDomainRepository.findTopicById(topicId);
                TopicModelView topicModelView =
copyTopicModelView(topic);
                SubjectModel subject =
this.topicDomainRepository.createSubjectDomain(null).findSubject(subject
Id);
```

```
                UserProblemModel userProblemModel = new
UserProblemModel();

 userProblemModel.setErrorCount(userProblem.getErrorCount());
                userProblemModel.setSubject(subject);
                userProblemModel.setTopic(topicModelView);
                userProblemModel.setProblemId(userProblem.getId());

 userProblemModel.setErrorAnswer(userProblem.getErrorAnswer());
                userProblemModel.setErrorTime(new
DateTime(userProblem.getErrorTime()).toString("yyyy-MM-dd HH:mm:ss"));
                userProblemModelList.add(userProblemModel);
            }
            ListPageModel<UserProblemModel> listPageModel = new
ListPageModel<>();
            listPageModel.setList(userProblemModelList);
            listPageModel.setPage(page);
            listPageModel.setPageCount(pageSize);
            listPageModel.setSize((int) userProblemListPage.getTotal());
            return CallResult.success(listPageModel);
        }
        Page<UserProblem> userProblemListPage =
this.topicDomainRepository.createUserProblem(null).findUserProblemListBy
SubjectId(userId,
searchSubjectId,ErrorStatus.NO_SOLVE.getCode(),page,pageSize);
        List<UserProblemModel> userProblemModelList = new ArrayList<>();
        List<UserProblem> userProblemList =
userProblemListPage.getRecords();
        for (UserProblem userProblem : userProblemList){
            Long topicId = userProblem.getTopicId();
            Long subjectId = userProblem.getSubjectId();
            Topic topic =
this.topicDomainRepository.findTopicById(topicId);
            TopicModelView topicModelView = copyTopicModelView(topic);
            SubjectModel subject =
this.topicDomainRepository.createSubjectDomain(null).findSubject(subject
Id);
            UserProblemModel userProblemModel = new UserProblemModel();
            userProblemModel.setErrorCount(userProblem.getErrorCount());
            userProblemModel.setSubject(subject);
            userProblemModel.setTopic(topicModelView);
            userProblemModel.setProblemId(userProblem.getId());

 userProblemModel.setErrorAnswer(userProblem.getErrorAnswer());
```

```
            userProblemModel.setErrorTime(new
DateTime(userProblem.getErrorTime()).toString("yyyy-MM-dd HH:mm:ss"));
            userProblemModelList.add(userProblemModel);
        }
        ListPageModel<UserProblemModel> listPageModel = new
ListPageModel<>();
        listPageModel.setList(userProblemModelList);
        listPageModel.setPage(page);
        listPageModel.setPageCount(pageSize);
        listPageModel.setSize((int) userProblemListPage.getTotal());
        return CallResult.success(listPageModel);
    }
```

SubjectDomain:

```
public Long findSubjectByInfo(String subjectName, String subjectGrade,
String subjectTerm) {
        if ("全部".equals(subjectName)){
            subjectName = null;
        }
        if ("全部".equals(subjectGrade)){
            subjectGrade = null;
        }
        if ("全部".equals(subjectTerm)){
            subjectTerm = null;
        }
        return
this.subjectDomainRepository.findSubjectByInfo(subjectName,subjectGrade,
subjectTerm);
    }
```

```
public Long findSubjectByInfo(String subjectName, String subjectGrade,
String subjectTerm) {

        LambdaQueryWrapper<Subject> queryWrapper =
Wrappers.lambdaQuery();
        boolean isNull = true;
        if (StringUtils.isNotBlank(subjectName)){
            queryWrapper.eq(Subject::getSubjectName,subjectName);
            isNull = false;
        }
        if (StringUtils.isNotBlank(subjectGrade)){
            queryWrapper.eq(Subject::getSubjectGrade,subjectGrade);
            isNull = false;
```

```
        }
        if (StringUtils.isNotBlank(subjectTerm)){
            queryWrapper.eq(Subject::getSubjectTerm,subjectTerm);
            isNull = false;
        }
        if (isNull){
            return null;
        }
        queryWrapper.last("limit 1");
        Subject subject = subjectMapper.selectOne(queryWrapper);
        return subject == null ? null : subject.getId();
    }
```

UserProlemDomain:

```
 public Page<UserProblem> findUserProblemList(Long userId, int
errorStatus,int page, int pageSize) {
        return
userProblemDomainRepository.findUserProblemList(userId,errorStatus,page,
pageSize);
    }
 public Page<UserProblem> findUserProblemListBySubjectId(Long userId,
Long searchSubjectId, int errorStatus, int page, int pageSize) {
        return
userProblemDomainRepository.findUserProblemListBySubjectId(userId,search
SubjectId,errorStatus,page,pageSize);
    }
```

```
 public Page<UserProblem> findUserProblemList(Long userId, int
errorStatus, int page, int pageSize) {
        LambdaQueryWrapper<UserProblem> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserProblem::getUserId,userId);
        queryWrapper.eq(UserProblem::getErrorStatus,errorStatus);
        return userProblemMapper.selectPage(new Page<>(page,pageSize),
queryWrapper);
    }

    public Page<UserProblem> findUserProblemListBySubjectId(Long userId,
Long searchSubjectId, int errorStatus, int page, int pageSize) {
        LambdaQueryWrapper<UserProblem> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserProblem::getUserId,userId);
        queryWrapper.eq(UserProblem::getSubjectId,searchSubjectId);
```

```
        queryWrapper.eq(UserProblem::getErrorStatus,errorStatus);
        return userProblemMapper.selectPage(new Page<>(page,pageSize),
queryWrapper);
    }
```

4.4 测试