

开始学习

购买课程后，就可以点击开始学习，进入学习流程

1. 课程科目

用户需要先选择练习的科目以及单元，才能进行学习

接口：/api/course/subjectInfo

参数：{courseId: 9}

返回值：

```
{
  "code":2000000000,
  "message":"default success",
  "result":[
    {
      "id":1,
      "subjectName":"初一政治 七年级 上",
      "subjectGrade":"七年级",
      "subjectTerm":"上",
      "subjectUnitList":[
        1,
        2,
        3,
        4,
        5,
        6,
        7,
        8
      ],
      "subjectUnitSelectedList":null
    },
    {
```

```

        "id":2,
        "subjectName":"初一语文 七年级 下",
        "subjectGrade":"七年级",
        "subjectTerm":"下",
        "subjectUnitList":[

        ],
        "subjectUnitSelectedList":null
    },
    {
        "id":6,
        "subjectName":"历史 七年级 下",
        "subjectGrade":"七年级",
        "subjectTerm":"下",
        "subjectUnitList":[
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8
        ],
        "subjectUnitSelectedList":null
    }
],
"success":true
}

```

1.1 业务逻辑

1. 根据课程id查询课程所对应的科目列表
2. 根据科目查询对应的单元
3. 如果之前有学习记录，自动选择所选科目和单元
4. 科目需要显示 名称-年级-学期

1.2 代码

1.2.1 Controller

```
@RequestMapping("subjectInfo")
public CallResult subjectInfo(@RequestBody CourseParam courseParam){
    return courseService.subjectInfo(courseParam);
}
```

1.2.2 Service

```
package com.mszlu.xt.model.service;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.web.model.params.CourseParam;

public interface CourseService {

    CallResult subjectInfo(CourseParam courseParam);
}
```

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

1.2.3 Domain

```
public CallResult<Object> subjectInfo() {
    Long userId = UserThreadLocal.get();

    Long courseId = this.courseParam.getCourseId();
}
```

```

        Course course =
this.courseDomainRepository.findCourseById(courseId);
        if (course == null){
            return
        }
        CallResult.fail(BusinessCodeEnum.TOPIC_PARAM_ERROR.getCode(),"参数错误");

        List<SubjectModel> subjectList =
this.courseDomainRepository.createSubjectDomain(null).findSubjectModelLi
stByCourseId(courseId);
        List<SubjectViewModel> subjectModelList = new ArrayList<>();
        for (SubjectModel subject : subjectList){
            List<Integer> subjectUnitList =
this.courseDomainRepository.createSubjectDomain(null).findSubjectUnit(su
bject.getId());
            SubjectViewModel subjectViewModel = new SubjectViewModel();
            subjectViewModel.setId(subject.getId());
            subjectViewModel.setSubjectName(subject.getSubjectName()+"
"+subject.getSubjectGrade()+" "+subject.getSubjectTerm());
            subjectViewModel.setSubjectGrade(subject.getSubjectGrade());
            subjectViewModel.setSubjectTerm(subject.getSubjectTerm());
            subjectViewModel.setSubjectUnitList(subjectUnitList);

//            if (userId != null){
//                UserHistory userHistory =
this.courseDomainRepository.createTopicDomain(null).findUserHistory(user
Id, subject.getId(), HistoryStatus.NO_FINISH.getCode());
//                if (userHistory != null) {
//                    String subjectUnits =
userHistory.getSubjectUnits();
//                    if (StringUtils.isEmpty(subjectUnits)) {
//                        List<Integer> strings =
JSON.parseArray(subjectUnits, Integer.class);
//                        subjectViewModel.setSubjectUnitSelectedList(strings);
//                    }
//                }
//            }
            subjectModelList.add(subjectViewModel);
        }
        return CallResult.success(subjectModelList);
    }

```

```
public Course findCourseById(Long courseId) {  
    return courseMapper.selectById(courseId);  
}
```

```
public List<Integer> findSubjectUnit(Long subjectId) {  
  
    return subjectDomainRepository.findSubjectUnit(subjectId);  
}
```

```
public List<Integer> findSubjectUnit(Long subjectId) {  
    LambdaQueryWrapper<SubjectUnit> queryWrapper = new  
LambdaQueryWrapper<>();  
    queryWrapper.eq(SubjectUnit::getSubjectId, subjectId);  
    List<SubjectUnit> subjectUnits =  
subjectUnitMapper.selectList(queryWrapper);  
    return  
subjectUnits.stream().map(SubjectUnit::getSubjectUnit).collect(Collectors  
s.toList());  
}
```

Mapper:

```
package com.mszlu.xt.web.dao;  
  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.mszlu.xt.pojo.SubjectUnit;  
  
public interface SubjectUnitMapper extends BaseMapper<SubjectUnit> {  
}
```

1.3 测试

2. 练习

1. 进入初中后，七年级学生明明面对了许多“新变化”。为了重新塑造完美的自我形象，他下定决心改掉自己的坏习惯。理想中的我、重新塑造的我应该（ ） ①越来越有活力 ②全面接受他人的建议 ③越来越坚强 ④越来越上进

- ☐ A. ①③④
- ☐ B. ②③④
- ☐ C. ①②④
- ☐ D. ①②③

上一题

提交

统计信息

学科名称：道法 七年级 下	单元：1,2,3,4,5,6,7,8	开始时间：2021-10-09 01:07:12	题目总数：50
已答：0	未答：50	正确：0	错误：0

答题卡

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23

2.1 业务逻辑

- 1. 确保课程存在
- 2. 如果没有练习过，保存练习记录
- 3. 练习开始后，根据科目和单元随机50道题进行练习
- 4. 保存题目的完成情况，首次练习，返回第一套题的详情
- 5. 如果上次的练习未结束，从上次练习的习题位置继续
- 6. 需要记录进度，答对答错数目等信息

2.2 数据库设计

t_user_history: 用户练习历史记录

名	类型	长度	小数点	不是 n	虚拟	键	注释
id	bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	
user_id	bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
topic_total	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		题目总数
progress	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		进度
create_time	bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		创建时间
subject_units	varchar	255	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		练习单元
topic_pro	varchar	45	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		城市
history_status	tinyint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		状态
subject_id	bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		学科id
finish_time	bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		完成时间
error_count	int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		错误数量

```
package com.xiaopizhu.tiku.dao.data;
```

```
import lombok.Data;
```

```
@Data
```

```
public class UserHistory {
    private Long id;
    private Long userId;
    private Integer topicTotal;
    private Integer progress;
    private Long createTime;
    private String subjectUnits;
    private String topicPro;
    private Long subjectId;
    //1 未完成 2 已完成 3 已取消
    private Integer historyStatus;
    private Long finishTime;
    private Integer errorCount;
}
```

t_user_practice: 用户练习详情

分多少张表，也是根据数据进行测算，一般从100开始，假设分100张表，一张表存1千万的数据，100张表可以存10亿，基本能满足需求，也就是说这里我们选择存100张表即可。

如何保证用户的数据落在某张表内，这就需要一致性hash算法了，这里我们选择最简单的 $userId \% 100$ ，如果用户id=101，则 $101 \% 100 = 1$ 分在t_user_practice_1这张表中。

在实际开发中，我们以分10张表为例，使用的分表工具为shardingjdbc

分好的表：（注意中间的空格为_，电脑用的曲面屏，ui显示上有问题）

```
t user practice 0
t user practice 1
t user practice 2
t user practice 3
t user practice 4
t user practice 5
t user practice 6
t user practice 7
t user practice 8
t user practice 9
```

2.3 Api说明

接口：/api/topic/partice

参数：{subjectId: 1, subjectUnitList: [1, 2, 3, 4, 5, 6, 7, 8]}

返回值：

```
{
  "code":2000000000,
  "message":"default success",
  "result":{
    "topic":{
      "id":1438886847374741505,
      "topicTitle":"深圳对驾驶机动车礼让斑马线提出规范标准，机动车辆通过斑马线如没有减速让行属于违法。这一规定体现了（   ）\n①生命是宝贵的又是脆弱的，需要呵护\n②生命是坚韧和顽强的\n③礼让行人是社会文明进步的体现\n④对生命的尊重和敬畏",
      "topicType":1,
      "topicImgList":[
      ],
      "topicStar":1,
```

```
"topicAreaPro":"","  
"topicAreaCity":"天津",  
"fillBlankAnswer":null,  
"fillBlankTopicChoice":0,  
"radioChoice": [  
  {  
    "A": {  
      "content": "①②③",  
      "imageList": [  
        ]  
      }  
    },  
    {  
      "B": {  
        "content": "①②④",  
        "imageList": [  
          ]  
        }  
      },  
      {  
        "C": {  
          "content": "①③④",  
          "imageList": [  
            ]  
          }  
        },  
        {  
          "D": {  
            "content": "②③④",  
            "imageList": [  
              ]  
            }  
          }  
        ]  
      },  
      "mulChoice": null,  
      "answer": null,  
      "analyze": null,  
      "subject": null,  
      "lastUpdateTime": "2021-10-03",  
      "userAnswer": "",  
      "pstatus": 0
```

```
},
"total":50,
"practiceId":1446507337694068738,
"createTime":"2021-10-09 00:06:38",
"finishTime": "",
"useTime": "",
"subjectName":"道法 七年级 下",
"subjectUnitList":[
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8
],
"answered":0,
"trueNum":0,
"wrongNum":0,
"noAnswer":50,
"progress":1,
"topicAnswerStatusList":[
  {
    "topicId":1438886847374741505,
    "userAnswer": "",
    "pStatus":0
  },
  {
    "topicId":9669,
    "userAnswer": "",
    "pStatus":0
  },
  {
    "topicId":1438886847701897217,
    "userAnswer": "",
    "pStatus":0
  },
  {
    "topicId":1438886850822459394,
    "userAnswer": "",
    "pStatus":0
  },
  {
    "topicId":1438886851413856258,
```

```
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9508,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886832292024322,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9408,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9460,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9770,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886853351624707,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886840349282306,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886811677020161,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9569,
```

```
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886818320797698,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9529,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886804609617922,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886830345867265,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886820459892738,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886830794657794,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9538,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886829758664706,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9809,
```

```
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 9836,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 9719,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 1438886834028466178,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 1438886803493933057,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 1438886819423899651,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 1438886854983208964,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 1438886815456088067,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 1438886810385174530,
        "userAnswer": "",
        "pStatus": 0
    },
    {
        "topicId": 9729,
```

```
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886829628641281,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 9411,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886814617227267,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886801983983618,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886821688823810,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886806492860417,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886811035291651,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886835718770690,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886823257493508,
```

```
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886826445164546,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886833193799684,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886818190774273,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886815846158338,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886834808606722,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886840735158274,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886805310066689,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886845097234434,
    "userAnswer": "",
    "pStatus": 0
  },
  {
    "topicId": 1438886827883810817,
```



```

        "userAnswer":"","
        "pStatus":0
    }
}
},
"success":true
}

```

```

package com.mszlu.xt.web.model;

import lombok.Data;

import java.util.List;
import java.util.Map;

@Data
public class PracticeDetailModel {
    private TopicModelView topic;
    private Integer total;
    private Long practiceId;
    private String createTime;
    private String finishTime;
    private String useTime;
    private String subjectName;
    private List<Integer> subjectUnitList;
    private Integer answered;
    private Integer trueNum;
    private Integer wrongNum;
    private Integer noAnswer;
    private Integer progress;
    private List<Map<String, Object>> topicAnswerStatusList;

    public Integer getNoAnswer(){
        if (total != null & answered != null){
            return total - answered;
        }
        return total;
    }
}

```

```

package com.mszlu.xt.web.model;

```

```
import com.mszlu.common.model.topic.ContentAndImage;
import com.mszlu.common.model.topic.FillBlankChoice;
import lombok.Data;

import java.util.List;
import java.util.Map;

/**
 * @author Jarno
 */
@Data
public class TopicModelView {

    private Long id;

    private String topicTitle;

    private Integer topicType;

    private List<String> topicImgList;

    private Integer topicStar;

    private String topicAreaPro;

    private String topicAreaCity;

    private List<FillBlankChoice> fillBlankAnswer;

    private int fillBlankTopicChoice;

    private List<Map<String, ContentAndImage>> radioChoice;
    private List<Map<String, ContentAndImage>> mulChoice;

    private String answer;

    private String analyze;

    private Integer subject;

    private String lastUpdateTime;

    private String userAnswer;
    private Integer pStatus;
```

```
}
```

2.4 代码

2.4.1 Controller

```
package com.mszlu.xt.web.api;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.model.service.TopicService;
import com.mszlu.xt.web.model.params.TopicParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("topic")
public class TopicApi {

    @Autowired
    private TopicService topicService;

    @RequestMapping(value = "practice", method = RequestMethod.POST)
    public CallResult practice(@RequestBody TopicParam topicParam){
        return topicService.practice(topicParam);
    }
}
```

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

import lombok.Data;

import java.util.List;
```

```
/**
 * @author Jarno
 */
@Data
public class TopicParam {
    private Long id;
    private Integer topicType;
    private String topicTitle;
    private String topicImg;
    private String answer;
    private String topicAnalyze;
    private Long topicSubject;

    private Integer topicStar;

    private String topicAreaPro;

    private String topicAreaCity;

    private Integer page = 1;
    private Integer pageSize = 20;

    //subject
    private Long subjectId;

    private Long userId;
    //题目ID
    private Long topicId;
    //单元
    private Integer subjectUnit;
    //单元
    private List<Integer> subjectUnitList;

    private String subjectName;
    private String subjectGrade;
    private String subjectTerm;

    private Long practiceId;

    private Long courseId;
}
```

2.4.2 Service

```
package com.mszlu.xt.model.service;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.web.model.params.TopicParam;

public interface TopicService {
    /**
     * 开始学习
     * @param topicParam
     * @return
     */
    CallResult practice(TopicParam topicParam);
}
```

```
package com.mszlu.xt.web.service.impl;

import com.mszlu.common.model.CallResult;
import com.mszlu.common.service.AbstractTemplateAction;
import com.mszlu.xt.model.service.TopicService;
import com.mszlu.xt.web.domain.TopicDomain;
import com.mszlu.xt.web.domain.repository.TopicDomainRepository;
import com.mszlu.xt.web.model.params.TopicParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class TopicServiceImpl extends AbstractService implements
TopicService {

    @Autowired
    private TopicDomainRepository topicDomainRepository;
    @Override
    public CallResult practice(TopicParam topicParam) {
        TopicDomain topicDomain =
this.topicDomainRepository.createDomain(topicParam);
        return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> checkBiz() {
                return topicDomain.checkPracticeBiz();
            }
        });
    }
}
```

```

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

```

2.4.3 Domain

之前在选择课程科目的时候，需要判断此科目是否已经练习过且未完成，如果是，返回之前所选的单元。

```

public CallResult<Object> subjectInfo() {
    Long userId = UserThreadLocal.get();

    Long courseId = this.courseParam.getCourseId();
    Course course =
this.courseDomainRepository.findCourseById(courseId);
    if (course == null){
        return
CallResult.fail(BusinessCodeEnum.TOPIC_PARAM_ERROR.getCode(), "参数错误");
    }

    List<SubjectModel> subjectList =
this.courseDomainRepository.createSubjectDomain(null).findSubjectModelLi
stByCourseId(courseId);
    List<SubjectViewModel> subjectModelList = new ArrayList<>();
    for (SubjectModel subject : subjectList){
        List<Integer> subjectUnitList =
this.courseDomainRepository.createSubjectDomain(null).findSubjectUnit(su
bject.getId());
        SubjectViewModel subjectViewModel = new SubjectViewModel();
        subjectViewModel.setId(subject.getId());
        subjectViewModel.setSubjectName(subject.getSubjectName()+"
"+subject.getSubjectGrade()+" "+subject.getSubjectTerm());
        subjectViewModel.setSubjectGrade(subject.getSubjectGrade());
        subjectViewModel.setSubjectTerm(subject.getSubjectTerm());
        subjectViewModel.setSubjectUnitList(subjectUnitList);
    }
}

```

```

        if (userId != null){
            UserHistory userHistory =
this.courseDomainRepository.createHistoryDomain(null).findUserHistory(us
erId, subject.getId(), HistoryStatus.NO_FINISH.getCode());
            if (userHistory != null) {
                String subjectUnits = userHistory.getSubjectUnits();
                if (StringUtils.isEmpty(subjectUnits)) {
                    List<Integer> strings =
JSON.parseArray(subjectUnits, Integer.class);

subjectViewModel.setSubjectUnitSelectedList(strings);
                }
            }
        }
        subjectModelList.add(subjectViewModel);
    }
    return CallResult.success(subjectModelList);
}

```

```

package com.mszlu.xt.web.domain;

import com.mszlu.xt.pojo.UserHistory;
import com.mszlu.xt.web.domain.repository.UserHistoryDomainRepository;
import com.mszlu.xt.web.model.params.UserHistoryParam;

public class UserHistoryDomain {

    private UserHistoryDomainRepository userHistoryDomainRepository;
    private UserHistoryParam userHistoryParam;

    public UserHistoryDomain(UserHistoryDomainRepository
userHistoryDomainRepository, UserHistoryParam userHistoryParam) {
        this.userHistoryDomainRepository = userHistoryDomainRepository;
        this.userHistoryParam = userHistoryParam;
    }

    public UserHistory findUserHistory(Long userId, Long subjectId, int
historyStatus) {
        return
this.userHistoryDomainRepository.findUserHistory(userId, subjectId, histor
yStatus);
    }
}

```

```

        public UserHistory findUserHistoryById(Long userId, Long practiceId)
    {
        return
        this.userHistoryDomainRepository.findUserHistoryById(userId,practiceId);
    }
}

```

```

package com.mszlu.xt.web.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.mszlu.xt.pojo.UserHistory;
import com.mszlu.xt.web.dao.UserHistoryMapper;
import com.mszlu.xt.web.domain.UserHistoryDomain;
import com.mszlu.xt.web.model.params.UserHistoryParam;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class UserHistoryDomainRepository {
    @Resource
    private UserHistoryMapper userHistoryMapper;

    public UserHistoryDomain createDomain(UserHistoryParam
userHistoryParam) {
        return new UserHistoryDomain(this,userHistoryParam);
    }

    public UserHistory findUserHistory(Long userId, Long subjectId, int
historyStatus) {
        LambdaQueryWrapper<UserHistory> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(UserHistory::getUserId,userId);
        queryWrapper.eq(UserHistory::getSubjectId,subjectId);
        queryWrapper.eq(UserHistory::getHistoryStatus,historyStatus);
        queryWrapper.last("limit 1");
        return userHistoryMapper.selectOne(queryWrapper);
    }

    public UserHistory findUserHistoryById(Long userId, Long practiceId)
    {

```



```

        LambdaQueryWrapper<UserHistory> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(UserHistory::getUserId,userId);
        queryWrapper.eq(UserHistory::getId,practiceId);
        return userHistoryMapper.selectOne(queryWrapper);
    }
}

```

```

package com.mszlu.xt.web.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.UserHistory;

public interface UserHistoryMapper extends BaseMapper<UserHistory> {
}

```

如果已经练习过且未完成，接着上次练习的进度，同一个科目只能同时进行一个练习。

TopicDomain:

```

package com.mszlu.xt.web.domain;

import com.alibaba.fastjson.JSON;
import com.fasterxml.jackson.core.type.TypeReference;
import com.mszlu.common.enums.TopicType;
import com.mszlu.common.model.BusinessCodeEnum;
import com.mszlu.common.model.CallResult;
import com.mszlu.common.model.topic.ContentAndImage;
import com.mszlu.common.model.topic.FillBlankChoice;
import com.mszlu.common.utils.UserThreadLocal;
import com.mszlu.xt.pojo.Topic;
import com.mszlu.xt.pojo.UserHistory;
import com.mszlu.xt.pojo.UserPractice;
import com.mszlu.xt.web.dao.data.TopicDTO;
import com.mszlu.xt.web.domain.repository.TopicDomainRepository;
import com.mszlu.xt.web.model.PracticeDetailModel;
import com.mszlu.xt.web.model.SubjectModel;

```

```

import com.msztu.xt.web.model.TopicModelView;
import com.msztu.xt.web.model.enums.HistoryStatus;
import com.msztu.xt.web.model.params.TopicParam;
import org.apache.commons.lang3.StringUtils;
import org.joda.time.DateTime;
import org.springframework.beans.BeanUtils;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Map;

public class TopicDomain {

    private TopicDomainRepository topicDomainRepository;
    private TopicParam topicParam;

    public TopicDomain(TopicDomainRepository topicDomainRepository,
TopicParam topicParam) {
        this.topicDomainRepository = topicDomainRepository;
        this.topicParam = topicParam;
    }

    public CallResult<Object> checkPracticeBiz() {
        Long subjectId = topicParam.getSubjectId();
        Long userId = UserThreadLocal.get();
        List<Long> courseIdList =
this.topicDomainRepository.getCourseDomain(null).findCourseIdBySubject(s
ubjectId);
        if (courseIdList == null || courseIdList.size()<=0){
            return
CallResult.fail(BusinessCodeEnum.COURSE_NO_BUY.getCode(),"not buy
course");
        }

        Integer orderCount =
this.topicDomainRepository.getUserCourseDomain(null).countUserCourseByUs
erId(userId,courseIdList,System.currentTimeMillis());
        if (orderCount == null || orderCount == 0){
            return
CallResult.fail(BusinessCodeEnum.COURSE_NO_BUY.getCode(),"not buy
course");
        }

        return CallResult.success();
    }
}

```

```

public CallResult<Object> practice() {
    Long userId = UserThreadLocal.get();
    Long subjectId = topicParam.getSubjectId();
    String topicAreaPro = topicParam.getTopicAreaPro();
    Long practiceId = topicParam.getPracticeId();
    UserHistory userHistory = null;
    //如果是从我的学习中 直接进入, 根据学习记录id 进行查询
    //如果不是, 根据传递的学科id, 进行查询, 判断是否进行过学习, 上一次学习未完成
    不能开启下一次练习
    if (practiceId == null) {
        userHistory =
this.topicDomainRepository.createHistoryDomain(null).findUserHistory(use
rId, subjectId, HistoryStatus.NO_FINISH.getCode());
    }else{
        userHistory =
this.topicDomainRepository.createHistoryDomain(null).findUserHistoryById
(userId,practiceId);
    }
    if (userHistory == null){
        //新的练习
        return startNewStudy(subjectId,userId,topicAreaPro);
    }
    if (userHistory.getHistoryStatus() == 3){
        return
CallResult.fail(BusinessCodeEnum.PRACTICE_CANCEL.getCode(),BusinessCodeE
num.PRACTICE_CANCEL.getMsg());
    }
    //已有的练习题, 开始练习
    Integer progress = userHistory.getProgress();
    Long topicId =
this.topicDomainRepository.createUserPracticeDomain(null).findUserPracti
ce(userId,userHistory.getId(), progress);
    if (topicId == null){
        return CallResult.fail();
    }
    System.out.println("已有练习题: "+topicId);
    TopicDTO topic =
this.topicDomainRepository.findTopicAnswer(topicId,
userHistory.getId());
    PracticeDetailModel practiceModel = new PracticeDetailModel();
    practiceModel.setProgress(progress);
    practiceModel.setTotal(userHistory.getTopicTotal());
    practiceModel.setTopic(getTopicModelView(topic));
    practiceModel.setPracticeId(userHistory.getId());
    practiceModel.setPracticeId(userHistory.getId());
}

```

```

        int answered = userHistory.getProgress();
        if (answered != userHistory.getTopicTotal()){
            answered = userHistory.getProgress()-1;
        }
        practiceModel.setAnswered(answered);
        int trueNum =
this.topicDomainRepository.createUserPracticeDomain(null).countUserPracticeTrueNum(userId,userHistory.getId());
        int wrongNum =
this.topicDomainRepository.createUserPracticeDomain(null).countUserPracticeWrongNum(userId,userHistory.getId());
        practiceModel.setTrueNum(trueNum);
        practiceModel.setWrongNum(wrongNum);
        practiceModel.setNoAnswer(0);
        SubjectModel subject =
this.topicDomainRepository.createSubjectDomain(null).findSubject(userHistory.getSubjectId());
        practiceModel.setSubjectName(subject.getSubjectName()+"
"+subject.getSubjectGrade()+" "+subject.getSubjectTerm());
        List<Integer> subjectUnitList1 =
JSON.parseArray(userHistory.getSubjectUnits(), Integer.class);
        practiceModel.setSubjectUnitList(subjectUnitList1);
        practiceModel.setCreateTime(new
DateTime(userHistory.getCreateTime()).toString("yyyy-MM-dd HH:mm:ss"));
        practiceModel.setFinishTime(userHistory.getFinishTime() == 0 ?
"":new DateTime(userHistory.getFinishTime()).toString("yyyy-MM-dd
HH:mm:ss"));
        practiceModel.setUseTime(userHistory.getFinishTime() == 0 ?
"":useTime(userHistory.getFinishTime(),userHistory.getCreateTime()));
        List<Map<String,Object>> topicAnswerStatusList =
this.topicDomainRepository.createUserPracticeDomain(null).findUserPracticeAll(userId,userHistory.getId());
        practiceModel.setTopicAnswerStatusList(topicAnswerStatusList);
        return CallResult.success(practiceModel);
    }

    private CallResult startNewStudy(Long subjectId,Long userId,String
topicAreaPro) {
        List<Integer> subjectUnitList = topicParam.getSubjectUnitList();
        if (subjectUnitList == null){
            subjectUnitList = new ArrayList<>();
        }
        //随机50道题，开始新的练习

```

```

        List<Long> topicIdList =
this.topicDomainRepository.findTopicRandomList(subjectId,topicAreaPro,subjectUnitList);
        if (topicIdList.size() <= 0){
            return
        }
        CallResult.fail(BusinessCodeEnum.TOPIC_NO_PRACTICE.getCode(),"没有练习题,
        请正确选择科目,单元以及地区");

        UserHistory userHistory = new UserHistory();
        userHistory.setCreateTime(System.currentTimeMillis());
        userHistory.setHistoryStatus(HistoryStatus.NO_FINISH.getCode());
        userHistory.setSubjectId(subjectId);
        userHistory.setProgress(1);
        userHistory.setSubjectUnits(JSON.toJSONString(subjectUnitList));
        userHistory.setTopicTotal(topicIdList.size());
        if (StringUtils.isEmpty(topicAreaPro)){
            topicAreaPro = "全国";
        }
        userHistory.setTopicPro(topicAreaPro);
        userHistory.setUserId(userId);
        userHistory.setFinishTime(0L);
        userHistory.setErrorCount(0);

        this.topicDomainRepository.createHistoryDomain(null).saveUserHistory(userHistory);
        //用户的练习题
        for (Long topicId : topicIdList){
            UserPractice userPractice = new UserPractice();
            userPractice.setHistoryId(userHistory.getId());
            userPractice.setUserId(userId);
            userPractice.setTopicId(topicId);
            userPractice.setPStatus(0);
            userPractice.setUserAnswer("");

            this.topicDomainRepository.createUserPracticeDomain(null).saveUserPractice(userPractice);
        }

        TopicDTO topic =
this.topicDomainRepository.findTopicAnswer(topicIdList.get(0),
        userHistory.getId());
        TopicModelView topicModelView = getTopicModelView(topic);
        PracticeDetailModel practiceModel = new PracticeDetailModel();
        practiceModel.setTotal(userHistory.getTopicTotal());
    
```

```

        practiceModel.setPracticeId(userHistory.getId());
        int answered = userHistory.getProgress();
        if (answered != userHistory.getTopicTotal()){
            answered = userHistory.getProgress()-1;
        }
        practiceModel.setAnswered(answered);
        practiceModel.setPracticeId(userHistory.getId());
        int trueNum =
this.topicDomainRepository.createUserPracticeDomain(null).countUserPracticeTrueNum(userHistory.getId());
        int wrongNum =
this.topicDomainRepository.createUserPracticeDomain(null).countUserPracticeWrongNum(userHistory.getId());
        practiceModel.setTrueNum(trueNum);
        practiceModel.setWrongNum(wrongNum);
        practiceModel.setNoAnswer(0);
        SubjectModel subject =
this.topicDomainRepository.createSubjectDomain(null).findSubject(userHistory.getSubjectId());
        practiceModel.setSubjectName(subject.getSubjectName()+"
"+subject.getSubjectGrade()+" "+subject.getSubjectTerm());
        List<Integer> subjectUnitList1 =
JSON.parseArray(userHistory.getSubjectUnits(), Integer.class);
        practiceModel.setSubjectUnitList(subjectUnitList1);
        practiceModel.setCreateTime(new
DateTime(userHistory.getCreateTime()).toString("yyyy-MM-dd HH:mm:ss"));
        practiceModel.setFinishTime(userHistory.getFinishTime() == 0 ?
"" : new DateTime(userHistory.getFinishTime()).toString("yyyy-MM-dd
HH:mm:ss"));
        practiceModel.setUseTime(userHistory.getFinishTime() == 0 ?
"" : useTime(userHistory.getFinishTime(),userHistory.getCreateTime()));
        practiceModel.setTopic(topicModelView);
        practiceModel.setProgress(1);
        List<Map<String, Object>> topicAnswerStatusList =
this.topicDomainRepository.createUserPracticeDomain(null).findUserPracticeAll(userId,userHistory.getId());
        practiceModel.setTopicAnswerStatusList(topicAnswerStatusList);
        return CallResult.success(practiceModel);
    }

    private String useTime(Long finishTime, Long createTime) {
        long diff = finishTime - createTime;
        long dayTime = 24 * 60 * 60 * 1000;
        String useTime = "";
        long day = diff / dayTime;

```

```

        if (day > 0){
            useTime = day + "天";
        }
        Calendar calendar = Calendar.getInstance();
        calendar.setTimeInMillis(finishTime);
        int finishHour = calendar.get(Calendar.HOUR_OF_DAY);
        int finishMinute = calendar.get(Calendar.MINUTE);
        int finishSecond = calendar.get(Calendar.SECOND);
        Calendar calendar1 = Calendar.getInstance();
        calendar1.setTimeInMillis(createTime);
        int createHour = calendar1.get(Calendar.HOUR_OF_DAY);
        int createMinute = calendar1.get(Calendar.MINUTE);
        int createSecond = calendar1.get(Calendar.SECOND);
        int diffHour = finishHour - createHour;
        if (diffHour < 0){
            diffHour = -diffHour;
        }
        if (diffHour < 10){
            useTime += "0"+diffHour + ":";
        }else {
            useTime += diffHour + ":";
        }
        int diffMinute = finishMinute - createMinute;
        if (diffMinute < 0){
            diffMinute = -diffMinute;
        }
        if (diffMinute < 10){
            useTime += "0"+diffMinute + ":";
        }else {
            useTime += diffMinute + ":";
        }
        int diffSecond = finishSecond - createSecond;
        if (diffSecond < 0){
            diffSecond = -diffSecond;
        }
        if (diffSecond < 10){
            useTime += "0"+ diffSecond;
        }else {
            useTime += String.valueOf(diffSecond);
        }
        return useTime;
    }

```

```

private TopicModelView copyView(TopicDTO topic){
    TopicModelView topicModel = new TopicModelView();

```

```

        if (topic == null){
            return null;
        }
        BeanUtils.copyProperties(topic,topicModel);
        return topicModel;
    }

    private TopicModelView getTopicModelView(TopicDTO topic) {
        TopicModelView topicModel = copyView(topic);
        if (topic == null){
            return null;
        }
        String topicImg = topic.getTopicImg();
        if (!StringUtils.isEmpty(topicImg)){
            List<String> topicImgList = JSON.parseArray(topicImg,
String.class);
            topicModel.setTopicImgList(topicImgList);
        }
        if (topic.getTopicType() == TopicType.FILL_BLANK.getCode()){
            List<FillBlankChoice> fillBlankChoiceList =
JSON.parseArray(topic.getTopicChoice(), FillBlankChoice.class);

            topicModel.setFillBlankTopicChoice(fillBlankChoiceList.size());
            topicModel.setFillBlankAnswer(fillBlankChoiceList);
            topicModel.setAnswer(null);
            String userAnswer = topic.getUserAnswer();
            if (!StringUtils.isEmpty(userAnswer)){
                String[] split = userAnswer.split("\\$#\\$");
                String sss = "";
                for (String s : split) {
                    sss += s + " ";
                }
                topicModel.setUserAnswer(sss);
            }
        }
        if (topic.getTopicType() == TopicType.RADIO.getCode()){
            List<Map<String, ContentAndImage>> list =
JSON.parseObject(topic.getTopicChoice(), new
com.alibaba.fastjson.TypeReference<List<Map<String, ContentAndImage>>>()
{}));

            topicModel.setRadioChoice(list);
        }
        if (topic.getTopicType() == TopicType.MUL_CHOICE.getCode()){

```



```

        List<Map<String,ContentAndImage>> list =
JSON.parseObject(topic.getTopicChoice(), new
com.alibaba.fastjson.TypeReference<List<Map<String,ContentAndImage>>>()
{}));

        topicModel.setMulChoice(list);
    }

    topicModel.setLastUpdateTime(new
DateTime(topic.getLastUpdateTime()).toString("yyyy-MM-dd"));
    return topicModel;
}
}

```

TopicDomainRepository:

```

package com.mszlu.xt.web.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.mszlu.xt.pojo.Topic;
import com.mszlu.xt.pojo.UserPractice;
import com.mszlu.xt.web.dao.TopicMapper;
import com.mszlu.xt.web.dao.data.TopicDTO;
import com.mszlu.xt.web.domain.*;
import com.mszlu.xt.web.model.params.*;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.util.List;

@Component
public class TopicDomainRepository {

    @Resource
    private TopicMapper topicMapper;

    @Autowired
    private CourseDomainRepository courseDomainRepository;
    @Autowired
    private UserCourseDomainRepository userCourseDomainRepository;
}

```

```

@Autowired
private UserHistoryDomainRepository userHistoryDomainRepository;
@Autowired
private UserPracticeDomainRepository userPracticeDomainRepository;
@Autowired
private SubjectDomainRepository subjectDomainRepository;

public TopicDomain createDomain(TopicParam topicParam) {
    return new TopicDomain(this,topicParam);
}

public CourseDomain getCourseDomain(CourseParam courseParam) {
    return courseDomainRepository.createDomain(courseParam);
}

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

public UserHistoryDomain createHistoryDomain(UserHistoryParam
userHistoryParam) {
    return
userHistoryDomainRepository.createDomain(userHistoryParam);
}

public List<Long> findTopicRandomList(Long subjectId,
                                     String topicAreaPro,
                                     List<Integer> subjectUnitList)
{
    LambdaQueryWrapper<Topic> queryWrapper = new
LambdaQueryWrapper<>();
    queryWrapper.eq(Topic::getTopicSubject,subjectId);
    if (StringUtils.isNotBlank(topicAreaPro)) {
        queryWrapper.likeRight(Topic::getTopicAreaPro,
topicAreaPro);
    }
    if (CollectionUtils.isNotEmpty(subjectUnitList)) {
        queryWrapper.in(Topic::getSubjectUnit, subjectUnitList);
    }
    queryWrapper.last("order by RAND() LIMIT 50");
    Object object = topicMapper.selectObjs(queryWrapper);
    return (List<Long>) object;
}

```

```

        public UserPracticeDomain createUserPracticeDomain(UserPracticeParam
userPracticeParam) {
            return
userPracticeDomainRepository.createDomain(userPracticeParam);
        }

        public TopicDTO findTopicAnswer(Long userId,Long topicId, Long
userHistoryId) {
            UserPractice userPractice =
this.userPracticeDomainRepository.createDomain(null).findUserPracticeByT
opicId(userId,topicId,userHistoryId);
            Topic topic = topicMapper.selectById(topicId);
            TopicDTO topicDTO = new TopicDTO();
            BeanUtils.copyProperties(topic,topicDTO);
            topicDTO.setPStatus(userPractice.getPStatus());
            topicDTO.setUserAnswer(userPractice.getUserAnswer());
            return topicDTO;
        }

        public SubjectDomain createSubjectDomain(SubjectParam subjectParam)
{
            return subjectDomainRepository.createDomain(subjectParam);
        }
    }
}

```

2.4.3.1 UserPracticeDomain

```

package com.mszlu.xt.web.domain;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.mszlu.xt.pojo.UserPractice;
import com.mszlu.xt.web.dao.UserPracticeMapper;
import com.mszlu.xt.web.domain.repository.UserPracticeDomainRepository;
import com.mszlu.xt.web.model.params.UserPracticeParam;

import java.util.List;
import java.util.Map;

public class UserPracticeDomain {

```

```

private UserPracticeDomainRepository userPracticeDomainRepository;
private UserPracticeParam userPracticeParam;

public UserPracticeDomain(UserPracticeDomainRepository
userPracticeDomainRepository, UserPracticeParam userPracticeParam) {
    this.userPracticeDomainRepository =
userPracticeDomainRepository;
    this.userPracticeParam = userPracticeParam;
}

public void saveUserPractice(UserPractice userPractice) {
    userPracticeDomainRepository.save(userPractice);
}

public int countUserPracticeTrueNum(Long userId, Long userHistoryId)
{
    return
userPracticeDomainRepository.countUserPracticeNumByStatus(userId, userHis
toryId, 2);
}

public int countUserPracticeWrongNum(Long userId, Long userHistoryId)
{
    //1 错误答案
    return
userPracticeDomainRepository.countUserPracticeNumByStatus(userId, userHis
toryId, 1);
}

public List<Map<String, Object>> findUserPracticeAll(Long userId,
Long userHistoryId) {
    return
userPracticeDomainRepository.findUserPracticeAnswerMap(userId, userHistor
yId);
}

public Long findUserPractice(Long userId, Long userHistoryId,
Integer progress) {
    return
userPracticeDomainRepository.findUserPracticeTopic(userId, userHistoryId,
progress);
}

```

```

        public UserPractice findUserPracticeByTopicId(Long userId, Long
topicId, Long userHistoryId) {
            return
userPracticeDomainRepository.findUserPracticeByTopicId(userId, topicId,
userHistoryId);
        }
    }
}

```

```

package com.mszlu.xt.web.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.mszlu.xt.pojo.UserPractice;
import com.mszlu.xt.web.dao.UserPracticeMapper;
import com.mszlu.xt.web.domain.UserPracticeDomain;
import com.mszlu.xt.web.model.params.UserPracticeParam;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.util.List;
import java.util.Map;

@Component
public class UserPracticeDomainRepository {

    @Resource
    private UserPracticeMapper userPracticeMapper;

    public UserPracticeDomain createDomain(UserPracticeParam
userPracticeParam){
        return new UserPracticeDomain(this,userPracticeParam);
    }

    public void save(UserPractice userPractice) {
        userPracticeMapper.insert(userPractice);
    }

    public int countUserPracticeNumByStatus(Long userId,Long
userHistoryId, int pStatus) {
        LambdaQueryWrapper<UserPractice> queryWrapper = new
LambdaQueryWrapper<>();

```

```

        queryWrapper.eq(UserPractice::getHistoryId,userHistoryId);
        queryWrapper.eq(UserPractice::getUserId,userId);
        //2 正确答案
        queryWrapper.eq(UserPractice::getPStatus,pStatus);
        return userPracticeMapper.selectCount(queryWrapper);
    }

    public List<Map<String, Object>> findUserPracticeAnswerMap(Long
userId, Long userHistoryId) {
        QueryWrapper<UserPractice> queryWrapper = Wrappers.query();
        queryWrapper.eq("history_id",userHistoryId);
        queryWrapper.eq("user_id",userId);
        queryWrapper.select("topic_id as topicId","p_status as
pStatus","user_answer as userAnswer");
        return userPracticeMapper.selectMaps(queryWrapper);
    }

    public Long findUserPracticeTopic(Long userId, Long userHistoryId,
Integer progress) {
        int pre = progress - 1;
        LambdaQueryWrapper<UserPractice> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserPractice::getUserId,userId);
        queryWrapper.eq(UserPractice::getHistoryId,userHistoryId);
        queryWrapper.select(UserPractice::getTopicId);
        queryWrapper.last("limit "+pre+",1");
        UserPractice userPractice =
userPracticeMapper.selectOne(queryWrapper);
        return userPractice.getTopicId();
    }

    public UserPractice findUserPracticeByTopicId(Long userId, Long
topicId, Long userHistoryId) {
        LambdaQueryWrapper<UserPractice> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserPractice::getUserId,userId);
        queryWrapper.eq(UserPractice::getHistoryId,userHistoryId);
        queryWrapper.eq(UserPractice::getTopicId,topicId);
        return userPracticeMapper.selectOne(queryWrapper);
    }
}

```

```
package com.mszlu.xt.web.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.UserPractice;

public interface UserPracticeMapper extends BaseMapper<UserPractice> {
}
```

2.4.3.2 UserCourseDomain

```
package com.mszlu.xt.web.domain;

import com.mszlu.xt.pojo.UserCourse;
import com.mszlu.xt.web.domain.repository.UserCourseDomainRepository;
import com.mszlu.xt.web.model.params.CourseParam;
import com.mszlu.xt.web.model.params.UserCourseParam;

import java.util.List;

public class UserCourseDomain {

    private UserCourseDomainRepository userCourseDomainRepository;
    private UserCourseParam courseParam;

    public UserCourseDomain(UserCourseDomainRepository
userCourseDomainRepository, UserCourseParam courseParam) {
        this.userCourseDomainRepository = userCourseDomainRepository;
        this.courseParam = courseParam;
    }

    public Integer countUserCourse(Long courseId) {
        return userCourseDomainRepository.countUserCourse(courseId);
    }

    public UserCourse findUserCourseByUserIdAndCourseId() {
        Long courseId = courseParam.getCourseId();
        Long userId = courseParam.getUserId();
        Long time = courseParam.getTime();
        return
userCourseDomainRepository.findUserCourseByUserIdAndCourseId(courseId,us
erId,time);
    }
}
```

```

        public Integer countUserCourseByUserId(Long userId,
                                                List<Long> courseIdList,
                                                long currentTime) {

            return
userCourseDomainRepository.countUserCourseByUserId(userId, courseIdList, c
urrentTime);
        }
    }
}

```

```

package com.mszlu.xt.web.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.mszlu.xt.pojo.UserCourse;
import com.mszlu.xt.web.dao.UserCourseMapper;
import com.mszlu.xt.web.domain.UserCourseDomain;
import com.mszlu.xt.web.model.params.CourseParam;
import com.mszlu.xt.web.model.params.UserCourseParam;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.util.List;

@Component
public class UserCourseDomainRepository {

    @Resource
    private UserCourseMapper userCourseMapper;

    public UserCourseDomain createDomain(UserCourseParam courseParam){
        return new UserCourseDomain(this, courseParam);
    }

    public Integer countUserCourse(Long courseId) {
        LambdaQueryWrapper<UserCourse> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(UserCourse::getCourseId, courseId);
        return userCourseMapper.selectCount(queryWrapper);
    }

    public UserCourse findUserCourseByUserIdAndCourseId(Long courseId,
Long userId, Long time) {

```



```

        LambdaQueryWrapper<UserCourse> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.eq(UserCourse::getCourseId, courseId);
        queryWrapper.eq(UserCourse::getUserId, userId);
        queryWrapper.gt(UserCourse::getExpireTime, time);
        UserCourse userCourse =
this.userCourseMapper.selectOne(queryWrapper);
        return userCourse;
    }

    public Integer countUserCourseByUserId(Long userId, List<Long>
courseIdList, long currentTime) {
        LambdaQueryWrapper<UserCourse> queryWrapper = new
LambdaQueryWrapper<>();
        queryWrapper.in(UserCourse::getCourseId, courseIdList);
        queryWrapper.eq(UserCourse::getUserId, userId);
        queryWrapper.gt(UserCourse::getExpireTime, currentTime);
        return this.userCourseMapper.selectCount(queryWrapper);
    }
}

```

2.4.3.3 UserHistoryDomain

```

package com.mszlu.xt.web.domain;

import com.mszlu.xt.pojo.UserHistory;
import com.mszlu.xt.web.domain.repository.UserHistoryDomainRepository;
import com.mszlu.xt.web.model.params.UserHistoryParam;

public class UserHistoryDomain {

    private UserHistoryDomainRepository userHistoryDomainRepository;
    private UserHistoryParam userHistoryParam;

    public UserHistoryDomain(UserHistoryDomainRepository
userHistoryDomainRepository, UserHistoryParam userHistoryParam) {
        this.userHistoryDomainRepository = userHistoryDomainRepository;
        this.userHistoryParam = userHistoryParam;
    }

    public UserHistory findUserHistory(Long userId, Long subjectId, int
historyStatus) {

```

```

        return
this.userHistoryDomainRepository.findUserHistory(userId,subjectId,histor
yStatus);
    }

    public UserHistory findUserHistoryById(Long userId, Long practiceId)
{
        return
this.userHistoryDomainRepository.findUserHistoryById(userId,practiceId);
    }

    public void saveUserHistory(UserHistory userHistory) {
        userHistoryDomainRepository.save(userHistory);
    }
}

```

```

package com.mszlu.xt.web.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.mszlu.xt.pojo.UserHistory;
import com.mszlu.xt.web.dao.UserHistoryMapper;
import com.mszlu.xt.web.domain.UserHistoryDomain;
import com.mszlu.xt.web.model.params.UserHistoryParam;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class UserHistoryDomainRepository {
    @Resource
    private UserHistoryMapper userHistoryMapper;

    public UserHistoryDomain createDomain(UserHistoryParam
userHistoryParam) {
        return new UserHistoryDomain(this,userHistoryParam);
    }

    public UserHistory findUserHistory(Long userId, Long subjectId, int
historyStatus) {
        LambdaQueryWrapper<UserHistory> queryWrapper = new
LambdaQueryWrapper<>();

```

```

        queryWrapper.eq(UserHistory::getUserId,userId);
        queryWrapper.eq(UserHistory::getSubjectId,subjectId);
        queryWrapper.eq(UserHistory::getHistoryStatus,historyStatus);
        queryWrapper.last("limit 1");
        return userHistoryMapper.selectOne(queryWrapper);
    }

    public UserHistory findUserHistoryById(Long userId, Long practiceId)
    {
        LambdaQueryWrapper<UserHistory> queryWrapper = new
        LambdaQueryWrapper<>();
        queryWrapper.eq(UserHistory::getUserId,userId);
        queryWrapper.eq(UserHistory::getId,practiceId);
        return userHistoryMapper.selectOne(queryWrapper);
    }

    public void save(UserHistory userHistory) {
        this.userHistoryMapper.insert(userHistory);
    }
}

```

2.4.3.4 CourseDomain

```

public List<Long> findCourseIdBySubject(Long subjectId) {

    return
    this.courseDomainRepository.findCourseIdBySubject(subjectId);
}

```

```

public List<Long> findCourseIdBySubject(Long subjectId) {
    LambdaQueryWrapper<CourseSubject> queryWrapper = new
    LambdaQueryWrapper<>();
    queryWrapper.eq(CourseSubject::getSubjectId,subjectId);
    queryWrapper.select(CourseSubject::getCourseId);
    List<CourseSubject> courseSubjects =
    courseSubjectMapper.selectList(queryWrapper);
    return
    courseSubjects.stream().map(CourseSubject::getCourseId).collect(Collectors.toList());
}

```

2.4.3.5 SubjectDomain

```
public SubjectModel findSubject(Long subjectId) {
    Subject subject = subjectDomainRepository.findById(subjectId);
    return copy(subject);
}
```

2.4.4 Mapper

```
package com.mszlu.xt.web.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.Topic;
import com.mszlu.xt.web.dao.data.TopicDTO;
import org.apache.ibatis.annotations.Param;

public interface TopicMapper extends BaseMapper<Topic> {

    TopicDTO findTopicAnswer(@Param("topicId")Long topicId,
                             @Param("userHistoryId") Long
userHistoryId);
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.mszlu.xt.web.dao.TopicMapper">

    <resultMap id="TopicMap" type="com.mszlu.xt.pojo.Topic">
        <id column="id" property="id" jdbcType="BIGINT" />
        <result column="add_admin" property="addAdmin"
jdbcType="VARCHAR" />
        <result column="topic_type" property="topicType"
jdbcType="INTEGER" />
        <result column="topic_title" property="topicTitle"
jdbcType="VARCHAR" />
        <result column="topic_img" property="topicImg"
jdbcType="VARCHAR" />
        <result column="topic_choice" property="topicChoice"
jdbcType="VARCHAR" />
        <result column="answer" property="topicAnswer"
jdbcType="VARCHAR" />
    </resultMap>

```

```

        <result column="topic_analyze" property="topicAnalyze"
jdbcType="VARCHAR" />
        <result column="topic_subject" property="topicSubject"
jdbcType="INTEGER" />
        <result column="topic_star" property="topicStar"
jdbcType="INTEGER" />
        <result column="topic_area_pro" property="topicAreaPro"
jdbcType="VARCHAR" />
        <result column="topic_area_city" property="topicAreaCity"
jdbcType="VARCHAR" />
        <result column="create_time" property="createTime"
jdbcType="BIGINT" />
        <result column="last_update_time" property="lastUpdateTime"
jdbcType="BIGINT" />
        <result column="subject_unit" property="subjectUnit"
jdbcType="INTEGER" />
    </resultMap>

    <resultMap id="TopicAnswerMap"
type="com.mszlu.xt.web.dao.data.TopicDT0" extends="TopicMap">
        <result column="p_status" property="pStatus" />
        <result column="user_answer" property="userAnswer" />
    </resultMap>

    <select id="findTopicAnswer" parameterType="java.util.Map"
resultMap="TopicAnswerMap">
        select t.*,up.p_status,up.user_answer from t_topic t,
t_user_practice up where t.id=up.topic_id and t.id=#{topicId} and
up.history_id=#{userHistoryId} LIMIT 1
    </select>

</mapper>

```

2.4.5 其他用到的类

```

package com.mszlu.xt.web.dao.data;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import lombok.Data;

/**

```

```
* @author Jarno
*/
@Data
public class TopicDTO {

    @TableId(type = IdType.ASSIGN_ID)
    private Long id;

    private String addAdmin;

    private String topicTitle;

    private Integer topicType;

    private String topicImg;

    private String topicChoice;

    private Integer topicStar;

    private String topicAreaPro;

    private String topicAreaCity;

    private String topicAnswer;

    private String topicAnalyze;

    private Long topicSubject;

    private Long createTime;

    private Long lastUpdateTime;

    private Integer subjectUnit;

    private Integer pStatus;

    private String userAnswer;

}
```

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

```
import java.util.HashMap;
import java.util.Map;

/**
 * @author Jarno
 */
public enum HistoryStatus {
    /**
     * look name
     */
    NO_FINISH(1, "未完成"),
    FINISHED(2, "已完成"),
    CANCEL(3, "已取消");

    private static final Map<Integer, HistoryStatus> CODE_MAP = new
HashMap<>(3);

    static{
        for(HistoryStatus topicType: values()){
            CODE_MAP.put(topicType.getCode(), topicType);
        }
    }

    /**
     * 根据code获取枚举值
     * @param code
     * @return
     */
    public static HistoryStatus valueOfCode(int code){
        return CODE_MAP.get(code);
    }

    private int code;
    private String msg;

    HistoryStatus(int code, String msg) {
        this.code = code;
        this.msg = msg;
    }

    public int getCode() {
        return code;
    }
}
```

```

    public void setCode(int code) {
        this.code = code;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }
}

```

2.5 分表实现

上述代码，`t_user_practice`数据并没有按照分表存储，需要加入分表配置

```

#分表配置
# 配置 t_user_practice 表规则
spring.shardingsphere.rules.sharding.tables.t_user_practice.actual-data-nodes=master.t_user_practice_${0..9}

# 配置分表策略
spring.shardingsphere.rules.sharding.tables.t_user_practice.table-strategy.standard.sharding-column=user_id
spring.shardingsphere.rules.sharding.tables.t_user_practice.table-strategy.standard.sharding-algorithm-name=table-inline

# 配置 分片算法
spring.shardingsphere.rules.sharding.sharding-algorithms.table-inline.type=INLINE
spring.shardingsphere.rules.sharding.sharding-algorithms.table-inline.props.algorithm-expression=t_user_practice_${user_id % 10}

```

注意，进行分表后，涉及到`t_user_practice`的查询 均需要加入查询条件`userId`，否则无法正确定位到对应的表,或者造成多次查询

3. 提交答案

3.1 接口说明

接口：/api/topic/submit

参数：{answer: "A", practiceId: 1446522579274702800, topicId: 1438886803695259600}

返回值：

```
{
  "code":2000000000,
  "message":"default success",
  "result":{
    "answerFlag":true,
    "topicType":1,
    "answer":"A",
    "finish":false
  },
  "success":true
}
```

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

import lombok.Data;

@Data
public class TopicSubmitModel {

    private boolean answerFlag;

    private Integer topicType;

    private String answer;

    private boolean finish;
```

}


3.2 业务逻辑

- 1. 根据答案，判断答案是否正确
- 2. 选择题，填空题，判断题都需要判断，问答题不需要
- 3. 如果回答错误，加入错题本，并标识错误
- 4. 回答正确，标识回答正确
- 5. 如果是最后一道题，结束整个练习
- 6. 不是最后一道题，更新进度

3.3 数据库设计

错题本：

t_user_problem

字段	索引	外键	触发器	选项	注释	SQL 预览				
名				类型	长度	小数点	不是 n	虚拟	键	注释
id				bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
user_id				bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
subject_id				bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
topic_id				bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		错题
error_count				int	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		错误数量
error_status				tinyint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		状态
error_answer				varchar	45	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		错误答案
error_time				bigint	0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>		错误时间

```
package com.mszlu.xt.pojo;

import lombok.Data;

@Data
public class UserProblem {
    private Long id;
```

```
private Long userId;

private Long subjectId;

private Long topicId;

private Integer errorCount;

private Integer errorStatus;

private String errorAnswer;

private Long errorTime;
}
```

3.4 编码

3.4.1 Controller

```
@PostMapping("submit")
public CallResult submit(@RequestBody TopicParam topicParam){
    return topicService.submit(topicParam);
}
```

3.4.2 Service

```
CallResult submit(TopicParam topicParam);
```

```
@Override
@Transactional
public CallResult submit(TopicParam topicParam) {
    TopicDomain topicDomain =
this.topicDomainRepository.createDomain(topicParam);
    return this.serviceTemplate.execute(new
AbstractTemplateAction<Object>() {
        @Override
        public CallResult<Object> checkBiz() {
            return topicDomain.checkSubmitBiz();
        }

        @Override
```

```

        public CallResult<Object> doAction() {
            return topicDomain.submit();
        }
    });
}

```

3.4.3 Domain

```

private Topic topic;
private UserHistory userHistory;
//业务检查
public CallResult<Object> checkSubmitBiz() {
    Long topicId = topicParam.getTopicId();
    Long practiceId = topicParam.getPracticeId();
    Long userId = UserThreadLocal.get();
    Topic topic = this.topicDomainRepository.findTopicById(topicId);
    if (topic == null){
        return
        CallResult.fail(BusinessCodeEnum.TOPIC_NOT_EXIST.getCode(),"topic not
        exist");
    }
    UserHistory userHistory =
    this.topicDomainRepository.createHistoryDomain(null).findUserHistoryById
    (userId,practiceId);
    if (userHistory == null){
        return
        CallResult.fail(BusinessCodeEnum.PRACTICE_NO_EXIST.getCode(),"练习题不存
        在");
    }
    if (userHistory.getHistoryStatus() == 2){
        return
        CallResult.fail(BusinessCodeEnum.PRACTICE_FINISHED.getCode(),BusinessCod
        eEnum.PRACTICE_FINISHED.getMsg());
    }
    if (userHistory.getHistoryStatus() == 3){
        return
        CallResult.fail(BusinessCodeEnum.PRACTICE_CANCEL.getCode(),BusinessCodeE
        num.PRACTICE_CANCEL.getMsg());
    }
    this.topic = topic;
    this.userHistory = userHistory;
    return CallResult.success();
}

```

//执行业务

```
public CallResult<Object> submit() {
    Long topicId = topicParam.getTopicId();
    Long userId = UserThreadLocal.get();
    String answer = topicParam.getAnswer();
    boolean finish = false;
    boolean answerFlag = false;
    //1. 填空题 选择题 判断题
    //2
    String topicAnswer = topic.getTopicAnswer();
    Integer topicType = topic.getTopicType();
    if (topicType == TopicType.FILL_BLANK.getCode()) {
        if (!StringUtils.isEmpty(answer)) {
            String[] str = answer.split("\\$#\\$");
            TopicDTO topicDTO = new TopicDTO();
            BeanUtils.copyProperties(topic, topicDTO);
            TopicModelView topicModel = getTopicModelView(topicDTO);
            if (str.length == topicModel.getFillBlankTopicChoice())
            {
                for (int i = 0; i < str.length; i++) {
                    FillBlankChoice anObject =
topicModel.getFillBlankAnswer().get(i);
                    log.info("提交答案:{}", str[i]);
                    log.info("正确答案:{}", anObject);
                    if (str[i].equals(anObject.getContent())) {
                        answerFlag = true;
                    } else {
                        answerFlag = false;
                        break;
                    }
                }
            }
        }
    }
    if (topicType == TopicType.RADIO.getCode()) {
        if (topicAnswer.equals(answer)) {
            answerFlag = true;
        }
    }
    if (topicType == TopicType.MUL_CHOICE.getCode()) {
        //数据库 A,B,C 前端传递的答案 A,B,C
        if (topicAnswer.equals(answer)) {
            answerFlag = true;
        }
    }
}
```

```

        if (topicType == TopicType.JUDGE.getCode()) {
            if (topicAnswer.equals(answer)) {
                answerFlag = true;
            }
        }
        UserHistoryDomain historyDomain =
this.topicDomainRepository.createUserHistoryDomain(null);
        UserProblemDomain userProblemDomain =
this.topicDomainRepository.createUserProblem(null);
        UserPracticeDomain userPracticeDomain =
this.topicDomainRepository.createUserPracticeDomain(null);

        if (!answerFlag) {
            //要加入错题本
//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);

                userProblem.setErrorTime(System.currentTimeMillis());
                userProblemDomain.saveUserProblem(userProblem);
            } else {

                userProblemDomain.updateUserProblemErrorCount(userId, topicId, answer);
            }

            userPracticeDomain.updateUserPractice(userHistory.getId(), topicId,
userId, answer, 1);
        } else {

            userPracticeDomain.updateUserPractice(userHistory.getId(), topicId,
userId, answer, 2);
        }
        Integer progress = userHistory.getProgress();
        if (progress >= 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;
    }
    TopicSubmitModel topicSubmitModel = new TopicSubmitModel();
    //答案是否正确
    topicSubmitModel.setAnswerFlag(answerFlag);
    topicSubmitModel.setFinish(finish);
    //答案 如果是填空题 并且有多个空 前端 传递过来的 第一个答案$$第二个
    答案...

    //给前端返回的是 此道题的正确答案
    topicSubmitModel.setAnswer(topicAnswer);
    if (topic.getTopicType() == TopicType.FILL_BLANK.getCode()){
        TopicDTO topicDTO = new TopicDTO();
        BeanUtils.copyProperties(topic, topicDTO);
        TopicModelView topicModel = getTopicModelView(topicDTO);
        StringBuilder fillBlankAnswer = new StringBuilder();
        for (FillBlankChoice fillBlankChoice :
topicModel.getFillBlankAnswer()){

            fillBlankAnswer.append(fillBlankChoice.getId()).append(".").append(fill
BlankChoice.getContent()).append(";");
        }
        topicSubmitModel.setAnswer(fillBlankAnswer.toString());
    }

    topicSubmitModel.setTopicType(topicType);
    return CallResult.success(topicSubmitModel);
}

```

3.4.3.1 UserProblemDomain

```

package com.mszlu.xt.web.domain;

import com.mszlu.xt.pojo.UserProblem;
import com.mszlu.xt.web.domain.repository.UserProblemDomainRepository;
import com.mszlu.xt.web.model.params.UserProblemParam;

public class UserProblemDomain {
    private UserProblemDomainRepository userProblemDomainRepository;
    private UserProblemParam userProblemParam;
}

```

```

    public UserProblemDomain(UserProblemDomainRepository
userProblemDomainRepository, UserProblemParam userProblemParam) {
        this.userProblemDomainRepository = userProblemDomainRepository;
        this.userProblemParam = userProblemParam;
    }

    public UserProblem getUserProblem(Long userId, Long topicId) {
        return
userProblemDomainRepository.getUserProblem(userId,topicId);
    }

    public void saveUserProblem(UserProblem userProblem) {
        userProblemDomainRepository.save(userProblem);
    }

    public void updateUserProblemErrorCount(Long userId, Long topicId,
String answer) {

        userProblemDomainRepository.updateUserProblemErrorCount(userId,topicId,
answer);
    }
}

```

```

package com.mszlu.xt.web.domain.repository;

import
com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.mszlu.xt.pojo.UserProblem;
import com.mszlu.xt.web.dao.UserProblemMapper;
import com.mszlu.xt.web.domain.UserProblemDomain;
import com.mszlu.xt.web.model.params.UserProblemParam;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class UserProblemDomainRepository {

    @Resource
    private UserProblemMapper userProblemMapper;
}

```



```

    public UserProblemDomain createDomain(UserProblemParam
userProblemParam) {
        return new UserProblemDomain(this,userProblemParam);
    }

    public UserProblem getUserProblem(Long userId, Long topicId) {
        LambdaQueryWrapper<UserProblem> queryWrapper =
Wrappers.lambdaQuery();
        queryWrapper.eq(UserProblem::getTopicId,topicId);
        queryWrapper.eq(UserProblem::getUserId,userId);
        queryWrapper.last("limit 1");
        return userProblemMapper.selectOne(queryWrapper);
    }

    public void save(UserProblem userProblem) {
        this.userProblemMapper.insert(userProblem);
    }

    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.set("error_count","error_count+1");
        update.set("answer",answer);
        userProblemMapper.update(null, update);
    }
}

```

```

package com.mszlu.xt.web.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.UserProblem;

public interface UserProblemMapper extends BaseMapper<UserProblem> {
}

```

3.4.3.2 UserHistoryDomain

```
public void updateUserHistoryErrorCount(Long userHistoryId) {

    userHistoryDomainRepository.updateUserHistoryErrorCount(userHistoryId);
}

    public void updateUserHistoryStatus(Long historyId, int
historyStatus, long finishTime) {

    userHistoryDomainRepository.updateUserHistoryStatus(historyId,historySt
atus,finishTime);
}

    public void updateUserHistoryProgress(Long historyId) {

userHistoryDomainRepository.updateUserHistoryProgress(historyId);
}
```

```
public void updateUserHistoryErrorCount(Long userHistoryId) {
    UpdateWrapper<UserHistory> update = Wrappers.update();
    update.eq("id",userHistoryId);
    update.set("error_count","error_count+1");
    this.userHistoryMapper.update(null, update);
}

    public void updateUserHistoryStatus(Long historyId, int
historyStatus, long finishTime) {
        LambdaUpdateWrapper<UserHistory> update =
Wrappers.lambdaUpdate();
        update.eq(UserHistory::getId,historyId);
        update.set(UserHistory::getHistoryStatus,historyStatus);
        update.set(UserHistory::getFinishTime,finishTime);
        this.userHistoryMapper.update(null, update);
}

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

3.4.3.3 UserPracticeDomain

```
public Long findUserPractice(Long userId, Long userHistoryId, Integer
progress) {
    return
userPracticeDomainRepository.findUserPracticeTopic(userId,userHistoryId,
progress);
}

public UserPractice findUserPracticeByTopicId(Long userId, Long
topicId, Long userHistoryId) {
    return
userPracticeDomainRepository.findUserPracticeByTopicId(userId, topicId,
userHistoryId);
}

public void updateUserPractice(Long userHistoryId, Long topicId,
Long userId, String answer, int pStatus) {

    userPracticeDomainRepository.updateUserPractice(userHistoryId,topicId,u
serId,answer,pStatus);
}
```

```
public Long findUserPracticeTopic(Long userId, Long userHistoryId,
Integer progress) {
    int pre = progress - 1;
    LambdaQueryWrapper<UserPractice> queryWrapper =
Wrappers.lambdaQuery();
    queryWrapper.eq(UserPractice::getUserId,userId);
    queryWrapper.eq(UserPractice::getHistoryId,userHistoryId);
    queryWrapper.select(UserPractice::getTopicId);
    queryWrapper.last("limit "+pre+",1");
    UserPractice userPractice =
userPracticeMapper.selectOne(queryWrapper);
    return userPractice.getTopicId();
}

public UserPractice findUserPracticeByTopicId(Long userId, Long
topicId, Long userHistoryId) {
    LambdaQueryWrapper<UserPractice> queryWrapper =
Wrappers.lambdaQuery();
    queryWrapper.eq(UserPractice::getUserId,userId);
    queryWrapper.eq(UserPractice::getHistoryId,userHistoryId);
    queryWrapper.eq(UserPractice::getTopicId,topicId);
}
```

```

        return userPracticeMapper.selectOne(queryWrapper);
    }

    public void updateUserPractice(Long userHistoryId, Long topicId,
Long userId, String answer, int pStatus) {

        LambdaUpdateWrapper<UserPractice> updateWrapper =
Wrappers.lambdaUpdate();
        updateWrapper
            .eq(UserPractice::getUserId,userId)
            .eq(UserPractice::getHistoryId,userHistoryId)
            .eq(UserPractice::getTopicId,topicId);
        updateWrapper.set(UserPractice::getUserAnswer,answer);
        updateWrapper.set(UserPractice::getPStatus,pStatus);
        userPracticeMapper.update(null, updateWrapper);
    }

```

3.4.4 Model

```

package com.mszlu.xt.web.model;

import com.mszlu.common.model.topic.FillBlankChoice;
import lombok.Data;

import java.util.List;
import java.util.Map;

/**
 * @author Jarno
 */
@Data
public class TopicModel {

    private String id;

    private String addAdmin;

    private String topicTitle;

    private String topicTypeStr;

    private Integer topicType;

    private String topicImg;

```

```

private String topicChoice;

private Integer topicStar;

private String topicAreaPro;

private String topicAreaCity;

private List<FillBlankChoice> fillBlankChoiceList;

private int fillBlankTopicChoice;

private List<Map<String,String>> radioChoice;
private List<Map<String,String>> mulChoice;

private String answer;

private String analyze;

private String subjectStr;

private Long subject;

private String createTime;

private String lastUpdateTime;
}

```

```

package com.mszlu.xt.web.model.enums;

import java.util.HashMap;
import java.util.Map;

/**
 * @author Jarno
 */
public enum ErrorStatus {
    /**
     * look name
     */
    NO_SOLVE(1,"未解决"),
    SOLVED(2,"已解决");
}

```

```
private static final Map<Integer, ErrorStatus> CODE_MAP = new  
HashMap<>(3);
```

```
static{  
    for(ErrorStatus topicType: values()){  
        CODE_MAP.put(topicType.getCode(), topicType);  
    }  
}
```

```
/**  
 * 根据code获取枚举值  
 * @param code  
 * @return  
 */  
public static ErrorStatus valueOfCode(int code){  
    return CODE_MAP.get(code);  
}
```

```
private int code;  
private String msg;
```

```
ErrorStatus(int code, String msg) {  
    this.code = code;  
    this.msg = msg;  
}
```

```
public int getCode() {  
    return code;  
}
```

```
public void setCode(int code) {  
    this.code = code;  
}
```

```
public String getMsg() {  
    return msg;  
}
```

```
public void setMsg(String msg) {  
    this.msg = msg;  
}
```

```
}
```

4. 跳转下一题

4.1 接口说明

接口：/api/topic/jump

参数：{page: 1, practiceId: "1446522579274702850"}

返回值：

```
{
  "code":2000000000,
  "message":"default success",
  "result":{
    "total":50,
    "progress":2,
    "topic":{
      "id":"1438886823773392898",
      "topicTitle":"小兵和小东是好朋友，小兵受外班同学欺负时，小东挺身而出帮他解围。小东数学有点偏科，为了保持年级前10位的名次，期末考试考数学的时候，小东要求数学学霸小兵把答案传递给他，小兵答应了。小兵的做法（   ）\n①正确，朋友之间理应互相帮助\n②正确，人应该学会感恩\n③错误，友谊不能没有原则\n④错误，竞争需要公平\n",
      "topicType":1,
      "topicImgList":[

    ],
      "topicStar":1,
      "topicAreaPro":"",
      "topicAreaCity":"遂宁",
      "fillBlankAnswer":null,
      "fillBlankTopicChoice":0,
      "radioChoice":[
        {
          "A":{
            "content":"①②",
            "imageList":[

          ]
        }
      ],
    },
  },
}
```

```
{
  "B":{
    "content":"②④",
    "imageList":[

    ]
  },
  {
    "C":{
      "content":"③④",
      "imageList":[

      ]
    },
    {
      "D":{
        "content":"②③",
        "imageList":[

        ]
      }
    }
  ],
  "mulChoice":null,
  "answer":null,
  "analyze":null,
  "subject":null,
  "lastUpdateTime":"2021-10-03",
  "userAnswer": "",
  "pstatus":0
},
"success":true
}
```


4.2 业务逻辑

1. 上面的提交答案，只是判断题的正确性，并更新进度，记录相关信息
2. 当提交之后，会进入下一题
3. 点击下一题，调用此接口，展示题目

4.3 编码

4.3.1 Controller

```
@PostMapping("jump")
public CallResult jump(@RequestBody TopicParam topicParam){
    return topicService.jump(topicParam);
}
```

4.3.2 Service

```
CallResult jump(TopicParam topicParam);
```

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

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

4.3.3 Domain

```
public CallResult<Object> jump() {
    Integer page = this.topicParam.getPage();
    if (page == null){
        return
CallResult.fail(BusinessCodeEnum.TOPIC_PARAM_ERROR.getCode(),"参数有误");
    }
    Long userId = UserThreadLocal.get();
```

```
        Long historyId = this.topicParam.getPracticeId();
        UserHistory userHistory =
this.topicDomainRepository.createUserHistoryDomain(null).findUserHistory
ById(historyId);
        Long topicId =
this.topicDomainRepository.createUserPracticeDomain(null).findUserPracti
ce(userId,userHistory.getId(),page);
        TopicDTO topic =
this.topicDomainRepository.findTopicAnswer(topicId,userId,userHistory.ge
tId());
        TopicModelView topicModelView = getTopicModelView(topic);
        Integer total = userHistory.getTopicTotal();
        Map<String,Object> map = new HashMap<>();
        map.put("total",total);
        map.put("progress",userHistory.getProgress());
        map.put("topic",topicModelView);
        return CallResult.success(map);
    }
```

