

# 第5天 题库展示

## 今日目标

- 掌握题库分类列表展示
- 掌握题目详细信息展示
- 掌握题目答案提交
- 掌握个人中心展示
- 了解小程序发布流程

## 1. 题库分类列表展示

### 1.1 功能分析

小程序默认题库列表，也称为刷题列表，这个列表也是我的错题本、收藏本、练习本的展示列表，这个题库首页列表主要是由技术点、企业、方向列表组成，效果如图所示：

技术点	企业	方向	技术点	企业	方向	技术点	企业	方向
HTML/CSS 3212题   已做11题		>	阿里巴巴 3212题   已做11题	名企	>	企业服务 3212题   已做11题		
Mysql 3212题   已做11题		>	腾讯 3212题   已做11题	名企	>	文化娱乐 3212题   已做11题		
Redis 3212题   已做11题		>	传智播客 3212题   已做11题	名企	>	数据服务 3212题   已做11题		
Sprint 3212题   已做11题		>	百度 3212题   已做11题	名企	>	在线教育 3212题   已做11题		
Vue.js 3212题   已做11题		>	小米 3212题   已做11题		>	智能硬件 3212题   已做11题		
HTML/CSS 3212题   已做11题		>	阿里巴巴 3212题   已做11题		>	企业服务 3212题   已做11题		
Mysql 3212题   已做11题		>	腾讯 3212题   已做11题		>	文化娱乐 3212题   已做11题		
Redis 3212题   已做11题		>	传智播客 3212题   已做11题		>	数据服务 3212题   已做11题		
Sprint 3212题   已做11题		>	百度 3212题   已做11题		>	在线教育 3212题   已做11题		

以上列表中的技术点是根据所选学科的学科目录来筛选，企业是根据当前所选城市的全部企业列表，方向是所选城市企业的所属行业方向。三种列表的展示内容来自数据库的不同数据表，但都可以进入题干展示。为便于从数据库提取数据和在不同列表切换数据方便，需要对这种展示数据进行一个数据定义，其中把按技术点、企业、方向三种类型的区分称为题库列表的种类（CategoryKind），三种列表（题库列表、错题本列表、收藏本列表、练习本列表）称为题库列表的类型（CategoryType）。

categoryKind  
1-按学科目录 2-按企业 3-按行业方向  
categoryType  
101-刷题、201-错题本、202-我的练习、203-收藏题目

每个列表需要显示该种类的总题目数量，还有显示当前用户已做题目的数量。用户做题数据是根据用户做题记录表(tr\_member\_question)来获取。

### 1.2 实现思路

根据业务分析，提取数据时需要识别当前用户身份，因为微信小程序开发环境不是标准浏览器模式，服务器无法用session+cookie的方式来完成会话状态，故需要客户端每次上传一个唯一标识来区分不同的客户。这个唯一标识也是最开始由服务端下发的，然后再同其他数据一起发送到服务端。在用户登录时，我们下发了一个token，这个token将作为用户的唯一标识。如图所示：

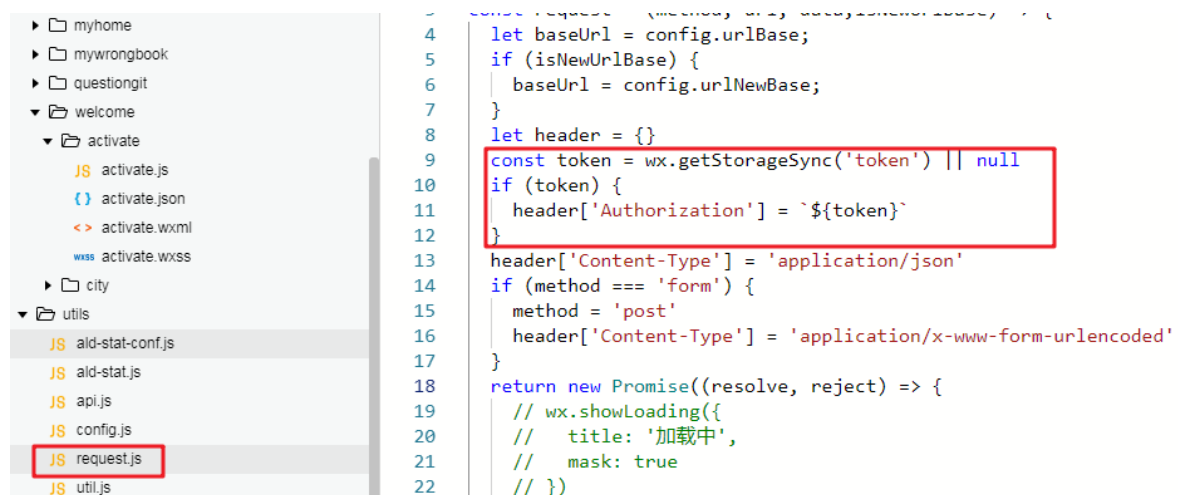
//返回微信用户信息

```
String wxUid = wxUser.getOpenId();  
//wxUser.setOpenId("");  
//wxUser.setUnionId("");  
Map map = new HashMap();  
map.put("token", wxUid);  
map.put("userInfo", wxUser);  
printResult(response, new Result(flag: true, message: "登录成功", map));
```

这个token，可以根据不同的策略来生成，在这里使用的是用户的openId，小程序拿到服务端返回的数据后，缓存到了小程序端，如图所示：



在每次发送请求时，客户端都在ajax请求中header中加入Authorization作为key，然后和数据一起发送到服务端，如图所示



服务端收到请求，在BaseController中，读取请求header的Authorization，如图如下：

```
public String getHeaderAuthorization(HttpServletRequest request) {  
    return request.getHeader(HEADER_AUTHORIZATION);  
}
```

综上所述，先从header中获取用户的唯一标识，然后根据这个标识获取userId，然后通过分类业务类（CategoryService）的业务方法，该方法根据userId、CategoryKind、CategoryType来获取分类列表。由于需要从三个不同的数据表中获取数据，可以先按技术点获取，整个流程走通后再实现另外两种分类数据的获取，只要实现其中一个，另外两个仅仅是Dao的不同，在这里实现按技术点获取。

## 1.3 按技术点获取

### 1.3.1 新增CatalogDao接口及映射文件

在列表中显示当前会员已完成题目数量是需要嵌套子查询来完成。

```
/**
 * @author : seanyang
 * @date : Created in 2019/8/18
 * @description : 学科目录Dao
 * @version: 1.0
 */
public interface CatalogDao {
    /**
     * 根据条件获取列表
     * @param queryParams(学科ID, 会员ID, 显示状态)
     * @return
     */
    List<Map> selectCatalogListByQueryParam(Map<String, Object> queryParams);
}
```

```
<?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.itheima.mm.wx.dao.CatalogDao">
    <select id="selectCatalogListByQueryParam" resultType="java.util.Map">
        select id,name as title ,
            (select count(*) from t_question where catalog_id = c.id) as
allCount,
            (select count(*) from t_question where catalog_id = c.id
            and id in (select question_id FROM tr_member_question where
member_id=#{memberId})) as finishedCount
        from t_catalog c
        WHERE course_id = #{courseId} and status = 0
    </select>
</mapper>
```

### 1.3.2 新增CategoryService接口及实现类

```
/**
 * @author : seanyang
 * @date : Created in 2019/8/18
 * @description : 首页业务接口
 * @version: 1.0
 */
public interface CategoryService {
    /**
     * 根据条件，获取首页题库分类列表
     * @param queryMap
     * @return
     */
}
```

```

    */
    List<Map> findCategory(Map<String, Object> queryMap);
}

```

```

@Component("categoryService")
@Slf4j
public class CategoryServiceImpl extends BaseService implements CategoryService
{
    @Override
    public List<Map> findCategory(Map<String, Object> queryMap) {
        log.info("findCategory query:{}", queryMap);
        Integer categoryKind = (Integer) queryMap.get("categoryKind");
        log.info("categoryKind:{}", categoryKind);
        List<Map> categoryList = null;
        if(categoryKind == QuestionConst.CategoryKind.CATALOG.getId()){
            // 查询以学科目录为主题的题目列表
            log.info("查询以学科目录为主题的题目列表");
            SqlSession sqlSession = SqlSessionUtils.openSession();
            CatalogDao catalogDao = sqlSession.getMapper(CatalogDao.class);
            categoryList = catalogDao.selectCatalogListByQueryParam(queryMap);
            sqlSession.close();
            return categoryList;
        }else if(categoryKind == QuestionConst.CategoryKind.COMPANY.getId()){
            // 查询以企业为主题分类的题目列表
            log.info("查询以企业为主题分类的题目列表");
            return categoryList;
        }else if(categoryKind == QuestionConst.CategoryKind.INDUSTRY.getId()){
            // 查询以分类为主题分类的题目列表
            log.info("查询以分类为主题分类的题目列表");
            return categoryList;
        }else{
            return new ArrayList<>();
        }
    }
}

```

### 1.3.3 新增CategoryController及服务调用

```

/**
 * @author : seanyang
 * @date : Created in 2019/8/18
 * @description : 分类控制器
 * 获取题库类型，对应首页列表分类显示
 * @version: 1.0
 */
@Component
@Slf4j
public class CategoryController extends BaseController {

    @HmSetter("categoryService")
    private CategoryService categoryService;

    @HmSetter("wxMemberService")

```

```

private wxMemberService wxMemberService;

/**
 * 获取题目不同分类列表
 * 按技术-1(按当前学科目录)、按企业-2、按方向（企业方向）-3
 * 刷题-101、错题本-201、我的练习202、收藏题目-203都是获取同类型列表
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
 */
@RequestMapping("/category/list")
public void getCategoryList (HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    try{
        Map<String,Object> paramData = parseJSON2Object(request,
HashMap.class);
        String openId = getHeaderAuthorization(request);
        // 根据openID,获取用户ID
        paramData.put("openId",openId);
        wxMember wxMember = wxMemberService.findByOpenId(openId);
        paramData.put("memberId",wxMember.getId());
        // 根据用户属性,获取用户选定的行业方向及城市
        paramData.put("courseId",wxMember.getCourseId());
        paramData.put("cityId",wxMember.getCityId());
        log.info("paramData:{",paramData);
        List<Map> mapList = categoryService.findCategory(paramData);
        printResult(response,new Result(true,"获取成功",mapList));
    }catch(RuntimeException e){
        log.error("getCategoryList",e);
        printResult(response,new Result(false,"获取失败"));
    }
}
}

```

### 1.3.4 测试接口

categoryList

POST http://localhost:7070/wx/category/list.do

Authorization Headers (2) Body Pre-request Script Tests

Key	Value
Content-Type	application/json
Authorization	oiu565SzoTCXctUD0y6LI-RQOkFg

New key Value

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "categoryKind": 1,
3   "categoryType": 101
4 }
```

分类种类、分类类型

Pretty Raw Preview JSON

```
1 {
2   "flag": true,
3   "message": "获取成功",
4   "result": [
5     {
6       "finishedCount": 7,
7       "id": 1,
8       "title": "Java基础",
9       "allCount": 150
10    },
11    {
12      "finishedCount": 0,
13      "id": 2,
14      "title": "JavaWeb",
15      "allCount": 226
16    }
17  ]
18 }
```

数据集

### 1.3.5 编写前端代码

分类列表页面是由pages/main组件来完成，main.js中提供了默认分类加载，如图所示：

```

onLoad: function (option) {
  var this = this
  _this.loadQuestions() // 加载问题列表
  _this.loadUserCenter() // 加载个人中心

  /* 业务请求
  */
  // 默认加载
  loadQuestions() {
    let _this = this

    let data = {
      categoryType: _this.data.currentTypeId,
      categoryKind: _this.data.currentKindId
    }

    app.api
      .questionsCategorys(data) // 调用api.questionsCategorys,传递data参数
      .then(res => {
        this.setData({
          questionList: res.data.result // 服务端返回数据,赋值给questionList
        })
      })
      .catch(res => {
        let errmsg = res.data.errmsg === undefined ? res.data : res.data.errmsg
        $Message({
          content: errmsg,

```

修订api.js中 questionsCategorys的定义，去掉原模拟实现，改为调用上述章节定义的分类列表API，如图所示：

```

const questionsCategorys = data => request('post', `/category/list.do`, data, true)

```

修订main.js中关于结果集赋值代码，如图所示：

```

app.api
  .questionsCategorys(data)
  .then(res => {
    this.setData({
      questionList: res.data.items // items改为result
    })
  })

```

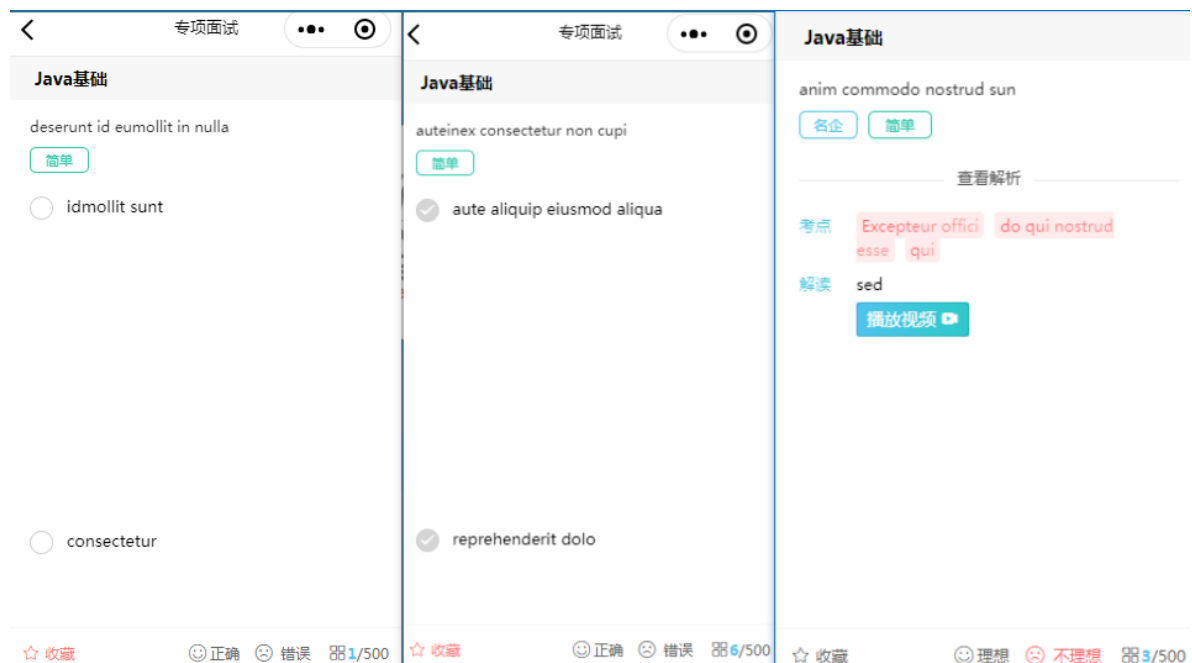
编译并运行小程序客户端，查看其效果：



## 2. 题目详情信息展示

### 2.1 功能分析

用户选择相应的分类后，点击进入题目详情。题目详情信息包含题干，选项（单选、复选，答案），解析内容、解析视频、难易度、是否名企、用户答案及当前所选分类的总题目数及用户已完成数量。



### 2.2 实现思路

题目详情是根据当前用户所选的分类种类、分类类型、分类ID，然后从数据库中读取相应的数据。数据读取采用一次性读取当前分类数据返回给前端（后续如果数据量非常大时再考虑分页获取）。另外在每个题目右下角都需要显示当前分类的总题目数量和已做数量，故可以考虑先根据分类ID获取当前分类数据，然后再获取这个分类的题目列表。



题目列表需要全部显示，同时需要显示是否是用户已完成的。所以需要题目表与用户题目关系表进行左或右连接。在查询题目数据的同时，需要关联查询其标签列表及题目选项列表。另外根据用户所选分类不同，获取题目列表的条件也不同。

整个查询中以小程序客户端需要的字段为基准，数据字段返回时可以使用别名方式命名，如果没有对应的POJO类，可以使用Map类封装数据。

本次查询内容比较多，可以先按技术点获取，整个流程走通后再实现另外两种分类数据的获取。另外两个仅仅是Dao的不同，在这里实现按技术点获取题目详情。

## 2.3 按技术点获取

### 2.3.1 更新Dao接口及映射文件

- 更新CatalogDao.xml文件

增加基于分类ID获取分类数据

```
<mapper namespace="com.itheima.mm.wx.dao.CatalogDao">
  <select id="selectCatalogListByQueryParam" resultType="java.util.Map">
    select id,name as title ,
      (select count(*) from t_question where catalog_id = c.id) as allCount,
      (select count(*) from t_question where catalog_id = c.id
        and id in (select question_id FROM tr_member_question where member_id=#{memberId})) as finishedCount
    from t_catalog c
    WHERE course_id = #{courseId} and status = 0
    <if test="categoryID != null ">
      and c.id = #{categoryID}
    </if>
  </select>
</mapper>
```

```
<if test="categoryID != null ">
  and c.id = #{categoryID}
</if>
```

- 新增QuestionDao接口

```
/**
 * 根据查询条件，获取题目列表
 * 根据分类类型 1-按学科目录 2-按企业 3-按行业方向
 * @param queryParams
 * @return
 */
List<Question> selectQuestionListByQueryParam(Map<String,Object>
queryParams);
```

- 新QuestionDao.xml文件

```
<?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.itheima.mm.wx.dao.QuestionDao">
  <resultMap id="baseResultMap" type="Question">
    <id property="id" column="id"/>
    <collection property="tags"
      column="id"
      select="selectTagsByQuestionId"/>
    <collection property="selection"
```

```

        column="id"
        select="selectQuestionItemById"/>
</resultMap>
<select id="selectQuestionListByQueryParam" resultMap="baseResultMap">
    SELECT * from (
        SELECT q.id,
        q.`subject` as title,
        difficulty as grade,
        analysis as content,
        analysis_video as video,
        '' as videoPoster,
        if(mq.tag is null,0,1) as isFinished,
        if(mq.is_favorite=1,1,0) as isFavorite,
        q.type,
        mq.tag as answerTag,
        q.catalog_id
        from t_question q LEFT JOIN tr_member_question mq
        on q.id = mq.question_id and mq.member_id = #{memberId} and status =
1
    ) subQuestion
    <where>
        <if test="categoryKind != null and categoryKind == 1 ">
            and catalog_id = #{categoryID}
        </if>
    </where>
</select>
<select id="selectTagsByQuestionId" resultType="Tag">
    select t.id,t.name as title
    from t_tag t
    where t.id in (select qt.tag_id from tr_question_tag qt where
qt.question_id = #{id})
</select>
<select id="selectQuestionItemById" resultType="QuestionItem">
    SELECT qi.id,'' as code ,qi.is_right as isRight,qi.content as title
    FROM t_question_item qi
    where qi.question_id = #{id}
</select>
</mapper>

```

### 2.3.2 更新CategoryService接口及实现类

```

/**
 * 根据条件，获取题库某一分类下的题目列表
 * @param queryMap
 * @return
 */
Map findCategoryQuestion(Map<String,Object> queryMap);

```

```

@Override
public Map findCategoryQuestion(Map<String, Object> queryMap) {
    log.info("findCategoryQuestion query:{})",queryMap);
    Integer categoryKind = (Integer) queryMap.get("categoryKind");
    Map resultMap = null ;
    if(categoryKind == QuestionConst.CategoryKind.CATALOG.getId()){

```

```

        log.info("查询以学科目录为主题的题目列表");
        SqlSession sqlSession = SqlSessionUtils.openSession();
        QuestionDao questionDao = sqlSession.getMapper(QuestionDao.class);
        CatalogDao catalogDao = sqlSession.getMapper(CatalogDao.class);
        // 根据条件获取某一学科目录的数据，复用上一节学科目录列表，加入基于学科目录ID条件
        // 此时结果集仅有一条记录
        List<Map> categoryList =
catalogDao.selectCatalogListByQueryParam(queryMap);
        if(categoryList != null && categoryList.size()>0 ){
            resultMap = categoryList.get(0);
            // 根据条件获取属于当前学科目录的题目列表，并放入获取的分类结果集
            List<Question> questionList =
questionDao.selectQuestionListByQueryParam(queryMap);
            resultMap.put("items",questionList);
        }
        sqlSession.close();
        return resultMap;
    }else if(categoryKind == QuestionConst.CategoryKind.COMPANY.getId()){
        // 查询以企业为主题分类的题目列表
        log.info(" 查询以企业为主题分类的题目列表");
        return resultMap;
    }else if(categoryKind == QuestionConst.CategoryKind.INDUSTRY.getId()){
        // 查询以分类为主题分类的题目列表
        log.info(" 查询以分类为主题分类的题目列表");
        return resultMap;
    }else{
        return new HashMap();
    }
}
}

```

### 2.3.3 更新CategoryController及服务调用

```

/**
 * 获取题目分类某一分类下的题目列表
 * 按技术-1(按当前学科目录)、按企业-2、按方向（企业方向）-3
 * 刷题-101、错题本-201、我的练习202、收藏题目-203都是获取同类型列表
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
 */
@RequestMapping("/category/question/list")
public void getCategoryQuestionList (HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
    try{
        Map<String,Object> paramData = parseJSON2Object(request, HashMap.class);
        String openId = getHeaderAuthorization(request);
        paramData.put("openId",openId);
        // 根据openId,获取用户ID
        wxMember wxMember = wxMemberService.findByOpenId(openId);
        paramData.put("memberId",wxMember.getId());
        // 根据用户属性，获取用户选定的行业方向及城市
        paramData.put("courseId",wxMember.getCourseId());
        paramData.put("cityId",wxMember.getCityId());
        log.info("paramData:{",paramData);
        Map resultMap = categoryService.findCategoryQuestion(paramData);
        printResult(response,new Result(true,"获取成功",resultMap));
    }
}

```

```

    }catch(RuntimeException e){
        log.error("getCategoryQuestionList",e);
        printResult(response,new Result(false,"获取失败"));
    }
}

```

### 2.3.4 测试接口

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

- Method:** POST
- URL:** http://localhost:7070/wx/category/question/list.do
- Body Type:** JSON (application/json)
- Request Body:**

```

{
  "categoryID": 1,
  "categoryKind": 1,
  "categoryType": 101
}

```
- Response Body:**

```

{
  "finishedCount": 7,
  "id": 1,
  "title": "Java基础",
  "allCount": 150,
  "items": [
    {
      "answerIsRight": true,
      "answerTag": 2,
      "content": "<p>springmvc执行流程：1.spring mvc把所有的请求都抛给DispatcherServlet查询一个或多个HandlerMapping,找到处理请求的Controller进行业务逻辑处理后，会返回一个ModelAndView<br />2.视图对象负责渲染返回给客户端。<br />3: mybatis触发条件：API接口层传递请求过来<br />4)根据SQL的ID查询"
    }
  ],
  "grade": 1,
  "id": 1,
  "isFamous": false,
  "isFavorite": true,
  "isFinished": true,
  "isLike": true
}

```

### 2.3.5 编写前端代码

在main.js中，从题库分类列表页跳转到题目详情页面：

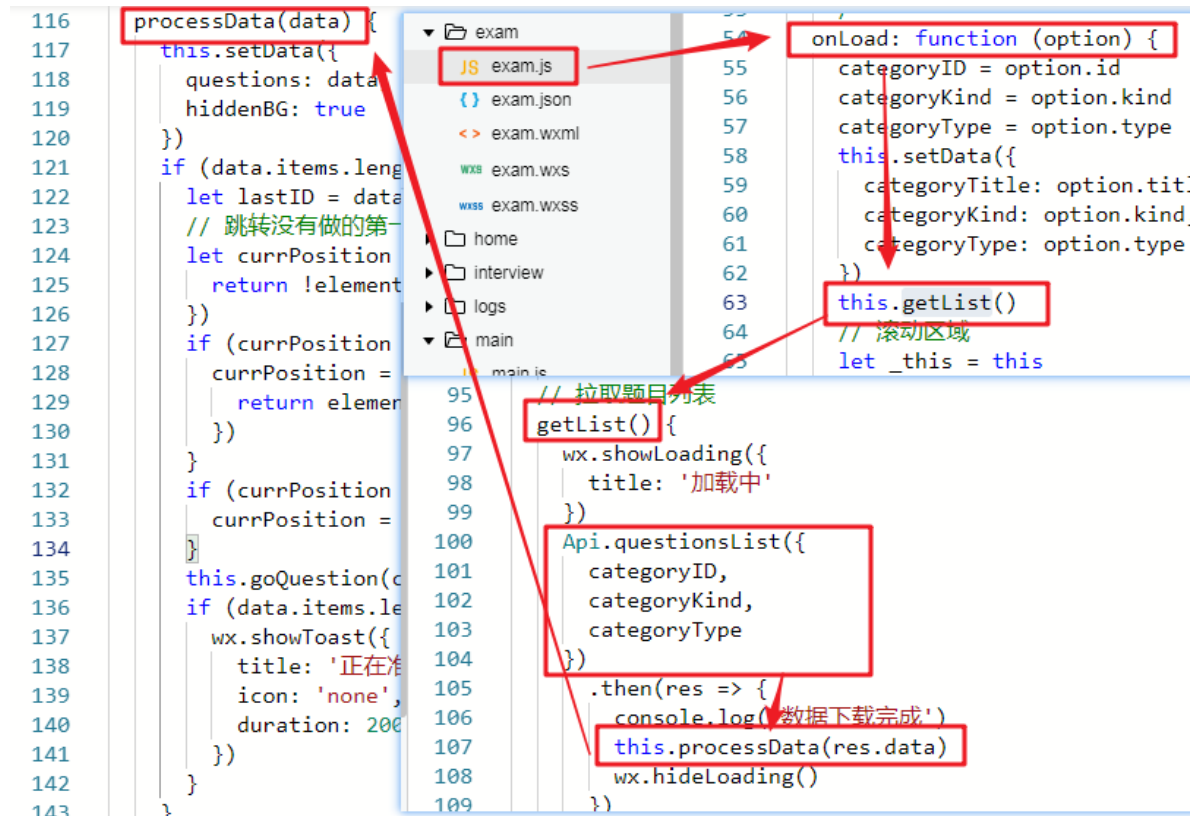
// 路由：去做题

```
handleExam: function (e) {
  let id = e.detail.id
  let title = e.detail.title

  let type = this.data.currentTypeId
  let kind = this.data.categoryKind

  wx.navigateTo({
    url: `../exam/exam?id=${id}&title=${title}&type=${type}&kind=${kind}`
  })
},
```

exam.js中，从onload开始，然后加载getList,然后是processData



按如图修改其中的前端代码：

api.js:

```
//const questionsList = data => request('get', `./questions/${data.categoryID}/${data.categoryKind}/${data.categoryType}`, data)
// 获取题库分类题目列表
const questionsList = data => request('post', `./category/question/list.do`, data, true)
```

```
const questionsList = data => request('post', `./category/question/list.do`, data, true)
```

exam.js:

```

onLoad: function (option) {
  //categoryID = option.id
  //categoryKind = option.kind
  //categoryType = option.type
  categoryID = parseInt(option.id)
  categoryKind = parseInt(option.kind)
  categoryType = parseInt(option.type)
  // ...
  103   Api.questionsList({
  104     categoryID,
  105     categoryKind,
  106     categoryType
  107   })
  108   .then(res => {
  109     console.log('数据下载完成')
  110     this.processData(res.data)
  111     wx.hideLoading()
  112   })
  113   .catch(err => {
  114     console.log(err)
  115     wx.hideLoading()
  116   })
}

```

把字符串改为数字类型

改为res.data.result

编译运行小程序，效果如图：



## 3. 题目答案提交

### 3.1 功能分析

在题目列表下发时，已经把每个题目的答案下发到客户端，故用户做题的答案验证是在客户端实现的。用户在切换下一题时把答题结果提交到服务器。当前题目类型有单选、复习及简答三种类型，做题的实现方式也会有所不同。

- 单选题目，只要用户点击任何一单选选项，页面立即响应答题结果
- 复选题目，用户选择题目选项，需要下拉到最下方提交答案，然后页面显示做题结果
- 简单题目，用户看到题目后，思考自己的题目答案，点击解析后查看是否与自己的思考匹配，选择下方的“理想”与“不理想”作为简答题的答案。

### 3.2 实现思路

单选与复选答案是正确与错误，简单题答案是理想与不理想，在数据库里面一个题目的答案就是正确与错误，需要在后端处理这两种情况，在答案提交时，也需要同时把当前收藏状态一并提交到服务端。单选、复习、简单在客户端已经完成了验证，只需要把题目答案提交到服务端即可。

单选与复选题目的答题结果，采用json数组字符串的形式存入数据库。

提交用户答案时，需要判断用户之前是否有做题记录，如果有，需要更新做题状态；没有就添加做题记录。

提交答案后，需要把当前做题的分类信息、最后一题信息更新到会员表中。

### 3.3 提交题目答案

#### 3.3.1 更新QuestionDao接口及映射文件

```
/**
 * 添加做题记录
 * @param paramsMap
 */
Integer addMemberQuestion(Map<String,Object> paramsMap);

/**
 * 更新做题记录
 * @param paramsMap
 */
Integer upateMemberQuestion(Map<String,Object> paramsMap);
```

```
<insert id="addMemberQuestion">
    INSERT INTO tr_member_question (member_id, question_id, is_favorite, tag,
answerResult)
    VALUES (#{memberId},#{id},#{isFavorite},#{answerTag},#{answerResult})
</insert>
<update id="upateMemberQuestion">
    UPDATE tr_member_question
    set is_favorite = #{isFavorite},tag = #{answerTag},answerResult=#{
answerResult}
    where member_id = #{memberId} and question_id = #{id}
</update>
```

#### 3.3.2 更新CategoryService接口及实现类

```

/**
 * 提交答案
 * @param mapMap
 */
void commitQuestion(Map<String, Object> mapMap);

```

```

@Override
public void commitQuestion(Map<String, Object> mapParams) {
    Integer type = (Integer) mapParams.get("type");
    // 回答的标志
    Integer answerTag = null;
    if( type == QuestionConst.Type.SINGLE.getId() || type ==
    QuestionConst.Type.MULTIPLE.getId()){
        // 单选题及选择题
        Boolean answerIsRight = (Boolean) mapParams.get("answerIsRight");
        answerTag = answerIsRight?
    QuestionConst.AnswerTag.PERFECT.ordinal():QuestionConst.AnswerTag.WRONG.ordinal(
    );

        String jsonStr = JSON.toJSONString(mapParams.get("answerResult"));
        mapParams.put("answerResult", jsonStr);
        mapParams.put("answerTag", answerTag);
    }else if(type == QuestionConst.Type.SIMPLE.getId()){
        // 简单题 判断理想、不理想
        Boolean answerIsRight = (Boolean) mapParams.get("answerIsRight");
        answerTag = answerIsRight?
    QuestionConst.AnswerTag.GOOD.ordinal():QuestionConst.AnswerTag.BAD.ordinal();
        mapParams.put("answerTag", answerTag);
    }
    // 收藏处理
    if( mapParams.get("isFavorite") != null){
        Boolean b1 = (Boolean) mapParams.get("isFavorite");
        mapParams.put("isFavorite", b1?1:0);
    }else{
        mapParams.put("isFavorite", 0);
    }

    log.info("add commitQuestion:{", mapParams);
    SqlSession sqlSession = SqlSessionUtils.openSession();
    QuestionDao questionDao = sqlSession.getMapper(QuestionDao.class);
    // 提交答题
    Integer result =questionDao.upateMemberQuestion(mapParams);
    if (result == 0){
        questionDao.addMemberQuestion(mapParams);
    }
    sqlSession.commit();
    sqlSession.close();
}

```

### 3.3.3 更新CategoryController及服务调用

```

/**
 * 提交题目
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException

```

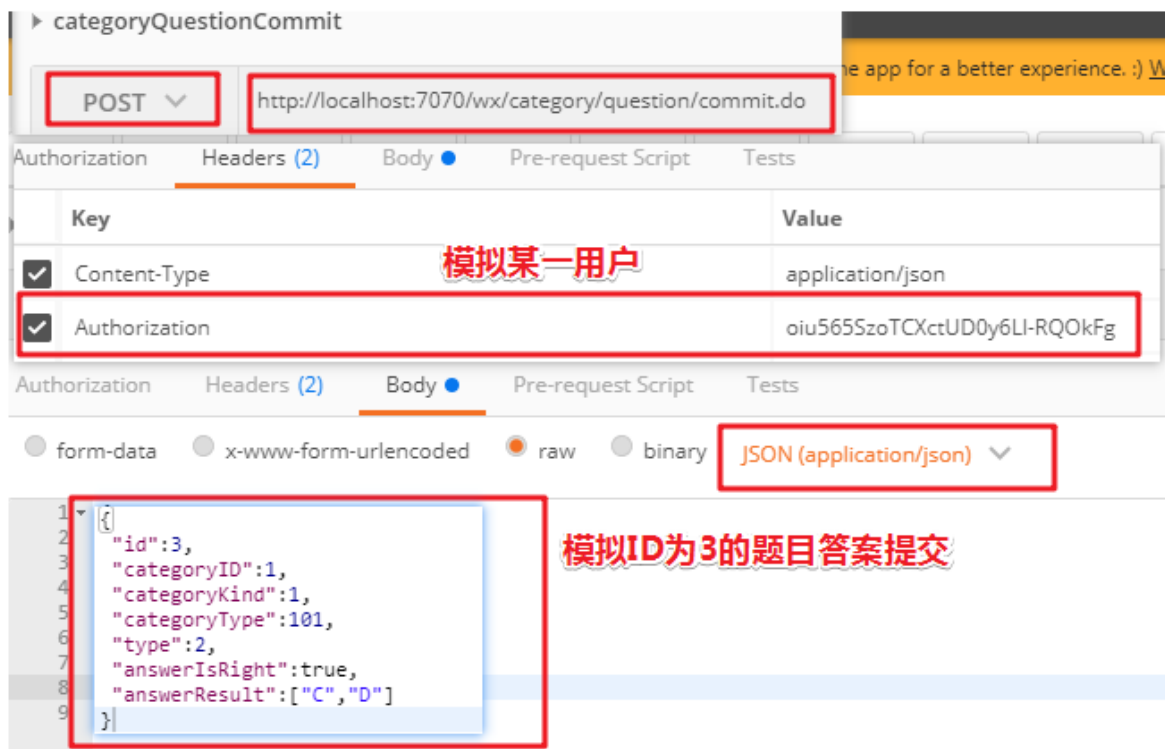


```

*/
@RequestMapping("/category/question/commit")
public void commitQuestion(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    try{
        Map<String,Object> paramData = parseJSON2Object(request, HashMap.class);
        log.info("commitQuestion paramData:{}",paramData);
        String openId = getHeaderAuthorization(request);
        paramData.put("openId",openId);
        // 根据openId,获取用户ID
        wxMember wxMember = wxMemberService.findByOpenId(openId);
        paramData.put("memberId",wxMember.getId());
        log.debug("paramData:{}",paramData);
        categoryService.commitQuestion(paramData);
        printResult(response,new Result(true,"提交成功"));
    }catch(RuntimeException e){
        log.error("commitQuestion",e);
        printResult(response,new Result(false,e.getMessage()));
    }
}
}

```

### 3.3.4 测试接口



### 3.3.5 编写前端代码

exam.js中的commentOne方法，在单选、复选及简答题提交时，都会调用此方法，方法如图所示：

```

// 提交单题
commentOne() {
  let currQuestion = this.data.questions.items
  [this.data.currPosition]
  Api.questionsCommitOne({
    categoryID,
    categoryKind,
    categoryType,
    id: currQuestion.id,
    type: currQuestion.type,
    answerIsRight: currQuestion.answerIsRight,
    answerResult: currQuestion.answerResult,
    isFavorite: currQuestion.isFavorite
  })
  .then(res => { })
  .catch(err => { })
},

```

修改api.js，调用后端答案提交接口，代码如下：

```

// 提交答案
//const questionsCommitOne = data => request('post', `/questions/${data.id}/${data.categoryID}/${data.categoryKind}/${data.categoryType}`, data)
const questionsCommitOne = data => request('post', `/category/question/commit.do`, data, true)

```

```

const questionsCommitOne = data => request('post',
  `/category/question/commit.do`, data, true)

```

先编译运行后端接口，再编译运行小程序客户端。为便于测试，可以清除数据库中用户做题记录，然后进行测试。清除数据后，已做题数据全部为0，如图所示：

技术点	企业	方向
Java基础 150题   已做 0题		>
JavaWeb 226题   已做 0题		>
Spring MVC 110题   已做 0题		>
Spring boot 105题   已做 0题		>

做题过程：

专项面试

### Java基础

<p>springmvc, mybatis的执行流程, spring中事物的管理</p>

简单

查看解析

考点 String字符串 数据类型与变量

解读 springmvc执行流程: 1.spring mvc将所有的请求都提交给DispatcherServlet,它会委托应用系统的其他模块负责对请求 进行真正的处理工作。  
2.DispatcherServlet查询一个或多个HandlerMapping,找到处理请求的Controller。  
3.DispatcherServlet将请求提交到目标Controller  
4.Controller进行业务逻辑处理后,会返回一个ModelAndView  
5.Dispathcher查询一个或多个ViewResolver视图解析器,找到ModelAndView对象指定的视图对象

简单题,用户自己点击理想与不理想

☆ 收藏 😊 理想 😞 不理想 88 1/150

### Java基础

下面关于线程创建的定义不正确的有 ( ) </p>

简单

✗ new Thread(){ public void run() { System.out.println("ThreadOne重写run方法!!"); }}

单选题目

✗ class ThreadOne extends Thread { public void run() { System.out.println("ThreadOne"); }}

点击选项后自动判断答案

☆ 收藏 😊 正确 😞 错误 88 3/150

### Java基础

关于递归的描述正确的是? ( ) </p>

简单

☒ 递归一定要有条件限定,保证递归能够停止下来

多选题目

确定按钮后,显示答题结果

确认答案

☆ 收藏 😊 正确 😞 错误 88 6/150

关于递归的描述正确的是? ( ) </p>

简单

✗ 递归的次数对程序的运行没有任何影响

✓ 递归一定要有条件限定,保证递归能够停止下来

答案

☆ 收藏 😊 正确 😞 错误 88 6/150

返回列表页面,查看已做题数量及数据库中的记录是否正确,结果如图所示:

技术点	企业	方向
Java基础		>
150题   已做 6题		

id	member_id	question_id	is_favorite	tag	answerResult
1	1	3	1	0	(Null)
3	1	5	1	1	(Null)
4	1	6	1	1	(Null)
5	1	7	1	0	(Null)
6	1	8	0	1	(Null)
7	1	9	0	2	(Null)
8	1	10	0	3	(Null)
45	6	1	0	2	(Null)
46	6	2	0	3	(Null)
47	6	3	0	1	["B"]
48	6	4	0	1	["A"]
49	6	5	0	0	["B"]
50	6	6	0	0	["D","A","B"]

### 3.4 更新用户做题信息

提交完答题记录之后，还需要更新会员最后的做题信息，便于从个人中心跳转继续做题。

#### 3.4.1 更新WxMemberDao接口及映射文件

```
/**
 * 更新成员最后做题信息
 * @param dataParams
 */
void updateMemberLastAnswer(Map<String, Object> dataParams);
```

```
<update id="updateMemberLastAnswer">
  update t_wx_member
  set last_category_kind = #{categoryKind}, last_category_type=#{categoryType},
  last_category_id=#{categoryID}, last_question_id=#{id}
  where open_id = #{openId}
</update>
```

#### 3.4.2 更新CategoryServiceImpl实现类

在commitQuestion方法中，加入更新用户做题信息的代码。

```
// 提交答题
Integer result = questionDao.updateMemberQuestion(mapParams);
if (result == 0) {
    questionDao.addMemberQuestion(mapParams);
}

// 更新会员，更新最后一题信息
memberDao = getDao(WxMemberDao.class);
memberDao.updateMemberLastAnswer(mapParams);
```

```
// 更新会员，更新最后一题信息
WxMemberDao memberDao = sqlSession.getMapper(WxMemberDao.class);
memberDao.updateMemberLastAnswer(mapParams);
```

### 3.4.3 编译并测试

先编译运行服务端，然后重启微信小程序，做题之前先看数据库会员信息表的状态，做题之后再去查看最新的状态，验证是否发生变化，正常如图所示：

	(Null)	(Null)	(Null)	(Null)
	1	100	1	7
做题之前	1	100	2	54
做题之后	3	100	1	7

## 4. 个人中心展示

个人中心需要显示当前用户的状态，状态包含当前用户已完成的题目数量及最后刷题的信息，然后可以根据这些信息跳转到上次最后做题的位置继续答题。返回数据中需要显示分类名称，故需要根据不同的参数获取不同的分类名称。

### 4.1 更新WxMemerDao接口及映射文件

```
/**
 * 获取会员中心数据
 * @param openId
 * @return
 */
Map<String, Object> selectCenter(String openId);

/**
 * 根据参数，获取名称
 * @param queryParams
 * @return
 */
String selectCategoryNameByQueryParams(Map<String, Object> queryParams);
```

```
<select id="selectCenter" resultType="java.util.Map">
    select
```

```

        (select count(*) from tr_member_question mq WHERE mq.member_id = m.id) as
answerCount,
        last_category_kind as categoryKind ,
        last_category_type as categoryType ,
        last_category_id as categoryID,
        last_question_id as lastQuestionId,
        city_id as cityID,
        (select data_value from t_dict where id = m.city_id) as cityName,
        course_id as subjectID
    from t_wx_member m
    WHERE m.open_id = #{openId}
</select>

<select id="selectCategoryNameByQueryParams" resultType="java.lang.String">

    <if test="categoryKind==1">
        select name as categoryTitle from t_catalog WHERE id = #{categoryID}
    </if>
    <if test="categoryKind==2">
        select short_name as categoryTitle from t_company WHERE id = #
{categoryID}
    </if>
    <if test="categoryKind==3">
        select name as categoryTitle from t_industry WHERE id = #{categoryID}
    </if>

</select>

```

## 4.2 更新WxMemberService接口及实现类

```

/**
 * 获取会员中心数据
 * @param openId
 * @return
 */
Map<String,Object> findMemberCenter(String openId);

```

```

@Override
public Map<String, Object> findMemberCenter(String openId) {
    SqlSession sqlSession = SqlSessionUtils.openSession();
    wxMemberDao wxMemberDao = sqlSession.getMapper(wxMemberDao.class);
    // 获取个人中心数据
    Map<String,Object> mapData = wxMemberDao.selectCenter(openId);
    String categoryTitle = "";
    // 获取查询类型名称
    if(mapData !=null && mapData.containsKey("categoryID")) {
        categoryTitle = wxMemberDao.selectCategoryNameByQueryParams(mapData);
    }
    mapData.put("categoryTitle",categoryTitle);
    sqlSession.close();
    return mapData;
}

```

## 4.3 更新WxMemberController及服务调用

```

@HmRequestMapping("/member/center")
public void getUserCenter (HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException{
    try{
        String openId = getHeaderAuthorization(request);
        log.info("getUserCenter openId:{}",openId);
        Map resultMap = wxMemberService.findMemberCenter(openId);
        printResult(response,new Result(true,"获取成功",resultMap));
    }catch(RuntimeException e){
        log.error("getUserCenter",e);
        printResult(response,new Result(false,"获取失败"));
    }
}

```

## 4.4 测试接口

The screenshot shows a REST client interface for a POST request to `http://localhost:7070/wx/member/center.do`. The **Headers** tab is selected, showing two headers: `Content-Type` with value `application/json` and `Authorization` with value `oiu565SzoTCXctUD0y6LI-RQOkFg`. The **Body** tab is also visible, showing a JSON response in `JSON` format:

```

{
  "flag": true,
  "message": "获取成功",
  "result": {
    "categoryType": 100,
    "categoryKind": 1,
    "answerCount": 1,
    "cityName": "上海",
    "categoryTitle": "Java基础",
    "cityID": 2,
    "categoryID": 1,
    "lastQuestionId": 7,
    "subjectID": 2
  }
}

```

## 4.5 编写前端代码

修改api.js，修改如下：

```

// 用户中心
// const userCenter = data => request('get', '/user/center', data)
const userCenter = data => request('get', '/member/center.do', data, true)

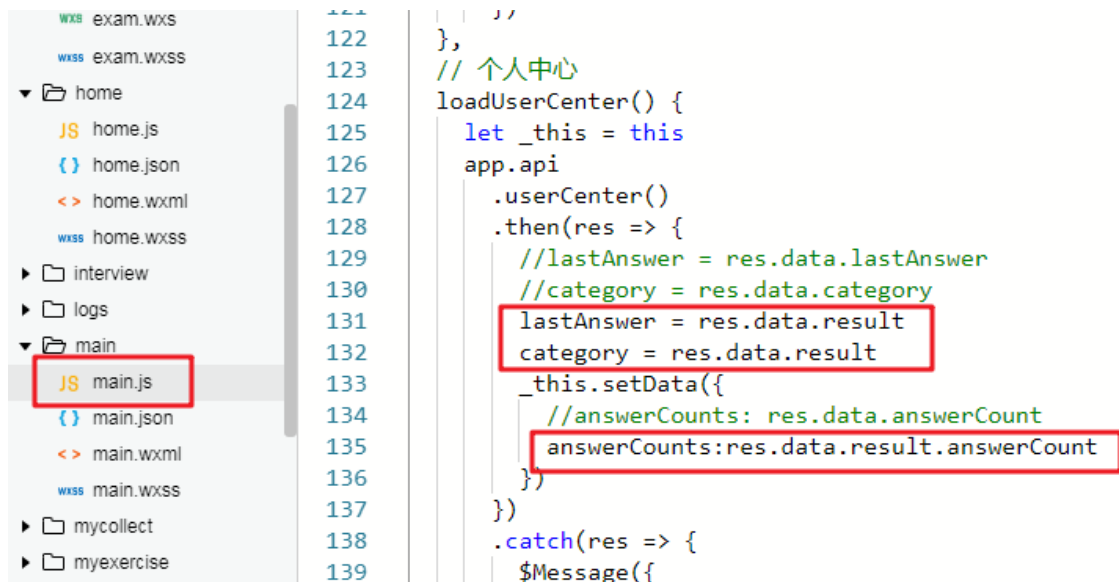
```

```

const userCenter = data => request('post', '/member/center.do', data, true)

```

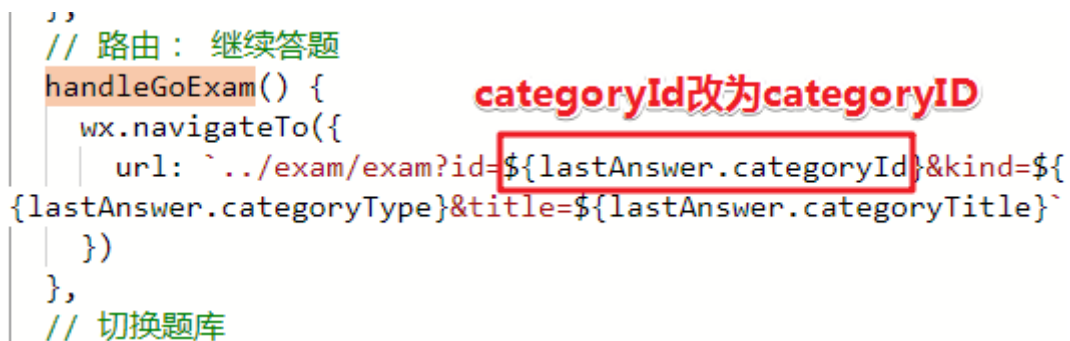
修改main.js，按如图修改：



效果如图：



继续答题改为：



## 5 发布微信小程序

发布小程序前，先修改为正式地址，然后通过工具上传到小程序管理后台，最后在小程序管理后台进行发布。

### 5.1 修改小程序的API地址

修改原本机调试地址，改为已配置好的域名地址：



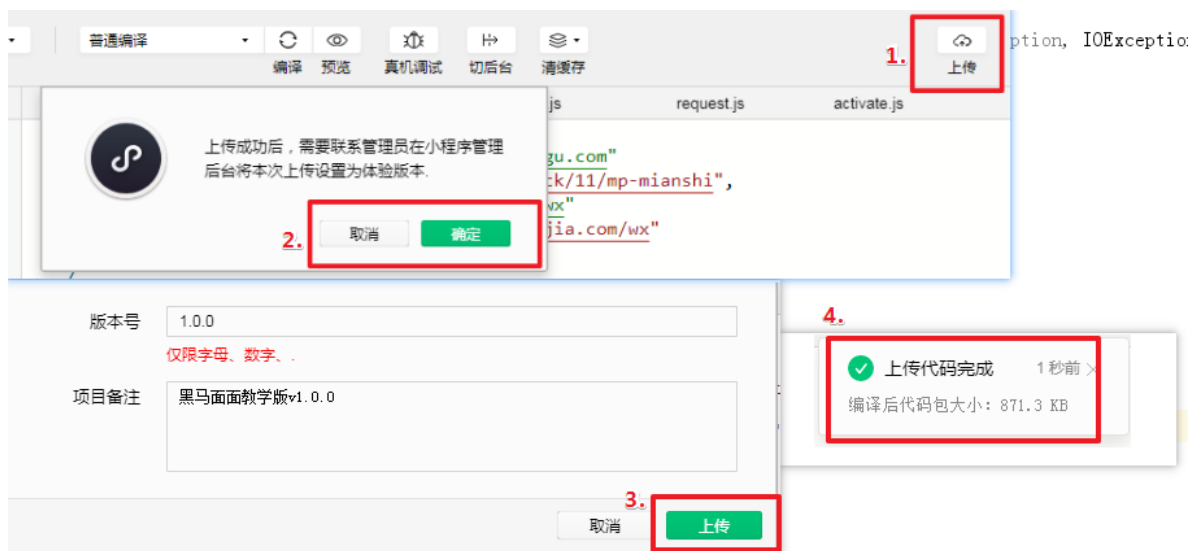


编译并在本机调试，查看效果，如图：



## 5.2 上传到小程序管理后台

测试OK后，上传到微信小程序管理后台，如图所示：



## 5.3 设置小程序基本信息

登录小程序管理后台，在首页有填写基本信息的入口，如果已填写完毕，如图所示



基本信息主要包含名称、简称及头像等信息，请根据实际情况如实填写：

基本设置		第三方设置	关联设置	关注公众号	违规记录
基本信息		说明			操作
小程序名称	黑马面面教学版	小程序发布前，可修改2次名称。当前还能修改2次。发布后，个人帐号可一年内修改2次名称。			修改
用户满意度	评价人数不足	来源于微信随机邀请的用户，结果将显示在小程序资料页			隐藏满意度
小程序简称	黑马面面	一年内可申请修改2次 今年还可修改2次			修改
小程序头像		一年内可申请修改5次 今年还可修改4次			修改
小程序码及线下物料下载		可下载小程序码及搜索框等线下推广物料			下载

## 5.4 开发设置

开发设置，主要设置域名，如图所示：

1

小程序信息

补充小程序的基本信息，如名称、图标、描述等

已完成

查看详情

小程序开发与管理

添加开发者

开发工具

下载开发者工具进行代码的开发和上传：[普通小程序开发者工具](#)、[小游戏开发者工具](#)

添加开发者

添加开发者，进行代码上传

配置服务器

在 [开发设置](#) 页面查看AppID和AppSecret，配置服务器域名

帮助文档

可以阅读入门介绍（[普通小程序](#) | [小游戏](#)）、开发文档（[普通小程序](#) | [小游戏](#)）、设计规范和 运营规范

## 服务器域名

服务器配置	说明	操作
request合法域名	https://mianshi. .	
socket合法域名	一个月内可申请5次修改 本月还可修改4次	修改
uploadFile合法域名		
downloadFile合法域名		
udp合法域名		

## 配置服务器信息

① 身份确认 — ② 配置服务器信息

服务器域名需经过ICP备案，新备案域名需24小时后方可配置。域名格式只支持英文大小写字母、数字及“-”，不支持IP地址。如果没有服务器与域名，可[前往腾讯云购买](#)。

request合法域名  +

## 5.5 发布小程序

从首页，点击“前往发布”

2

版本发布

先提交代码，然后提交审核，审核通过后可发布

前往发布

## 开发版本

版本号	开发者		提交审核
1.0.0	提交时间	2019-09-09 11:39:00	
	描述	黑马面面教学版v1.0.0	

点击提交审核，如图



### 确认提交审核？

提交给微信团队审核前，请确保：

- 提交的小程序功能完整，可正常打开和运行，而不是测试版或 Demo  
小程序的调试和预览可在开发者工具进行。多次提交测试内容或 Demo，将受到相应处罚。
- 已仔细阅读《微信小程序平台运营规范》和《微信小程序平台审核常见被拒绝情形》

☒ 已阅读并了解平台审核规则

下一步

取消



### 小程序内用户帐号登录规范调整提醒

为更好地保护用户隐私信息，优化用户体验，小程序需提供合理的用户帐号登录功能，请参照《小程序内用户帐号登录规范调整和优化建议》进行检查并做出调整。自2019年9月1日起，平台将会在小程序代码审核环节对不符合登录规范的帐号审核进行拦截。

取消

继续提交

配置功能页面

配置功能页面 至少填写一组，填写正确的信息有利于用户快速搜索出你的小程序

功能页面1

功能页面

pages/main/main

▼

标题

面试题库首页

12/32

所在服务类目

教育

▼

在线教育

▼

功能页面和服务类目必须一一对应，且功能页面提供的内容必须符合该类目范围

标签

面试题

标签用回车分开，填写与页面功能相关的标签，更容易被搜索

最后提交审核完成：


审核版本

版本号

1.0.0

审核中

开发者



提交审核时间

2019-09-09 12:59:00

描述

黑马面面教学版v1.0.0


详情

开发版本

版本号

1.0.0

开发者



提交时间

2019-09-09 11:39:00

描述

黑马面面教学版v1.0.0

提交审核

接下来等待腾讯审核结果即可，如何审核未通过，根据反馈进行修订，然后重新提交审核。