day15项目【微信支付】

# 01-课程支付功能需求描述

## 一、课程支付需求描述

### 1、课程支付说明

#### （1）课程分为免费课程和付费课程，如果是免费课程可以直接观看，如果是付费观看的课程，用户需下单支付后才可以观看



#### （2）如果是免费课程，在用户选择课程，进入到课程详情页面时候，直接显示 “立即观看”，用户点击立即观看，可以切换到播放列表进行视频播放

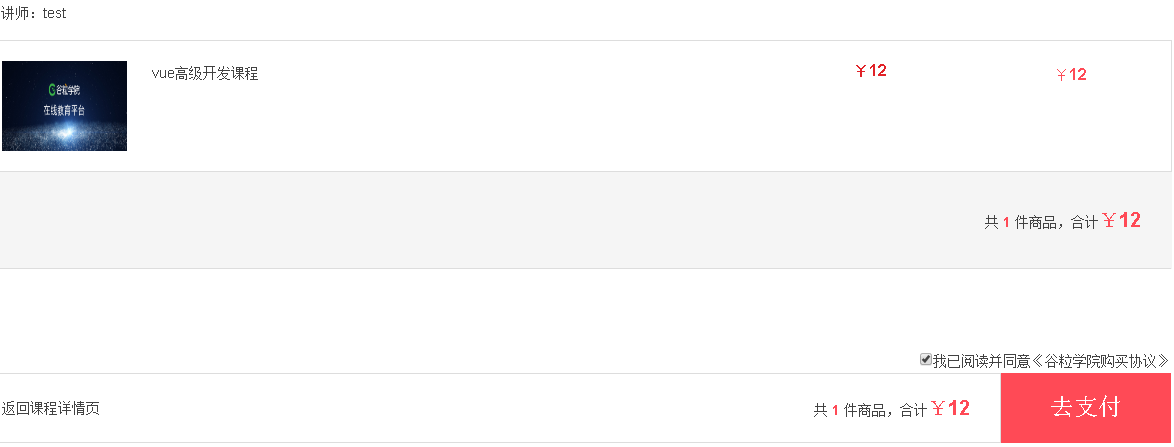


### 2、付费课程流程

#### （1）如果是付费课程，在用户选择课程，进入到课程详情页面时候，会显示 “立即购买”



#### （2）点击“立即购买”，会生成课程的订单，跳转到订单页面



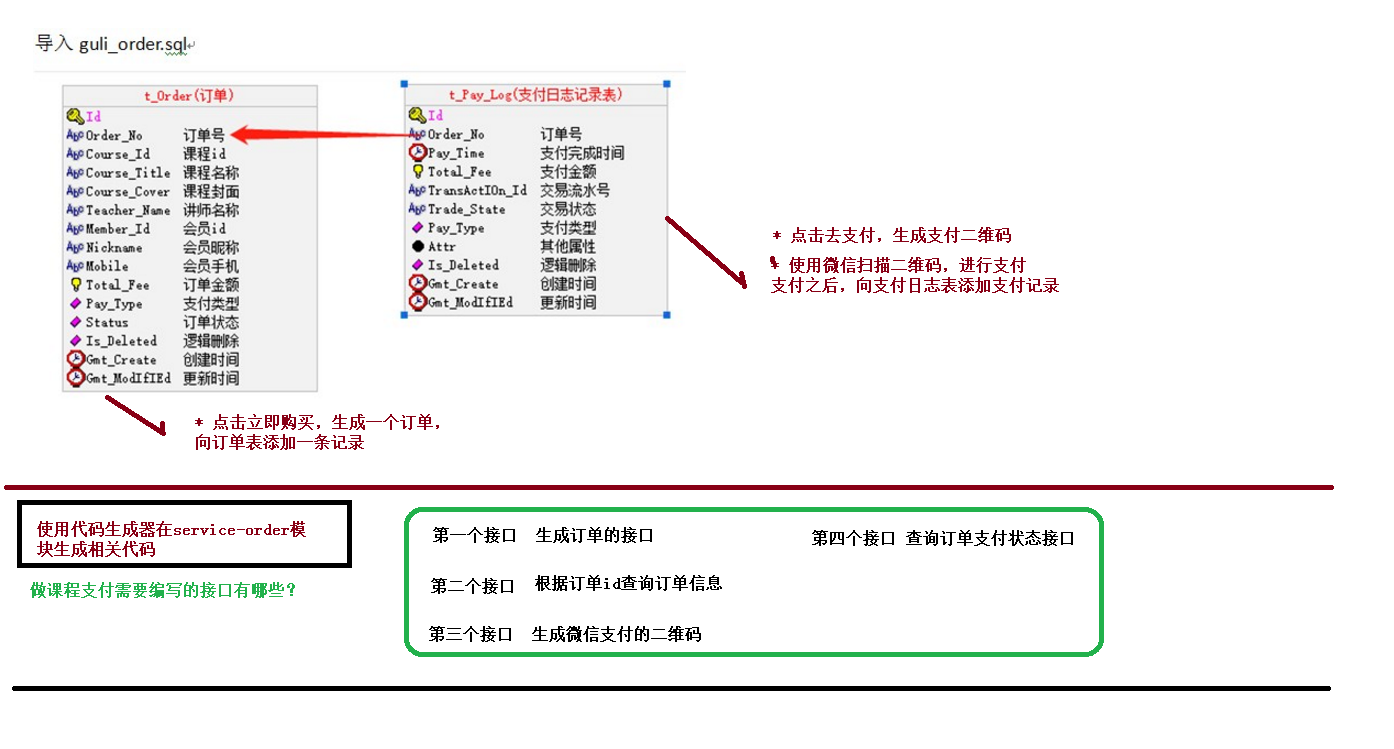
#### （3）点击“去支付”，会跳转到支付页面，生成微信扫描的二维码

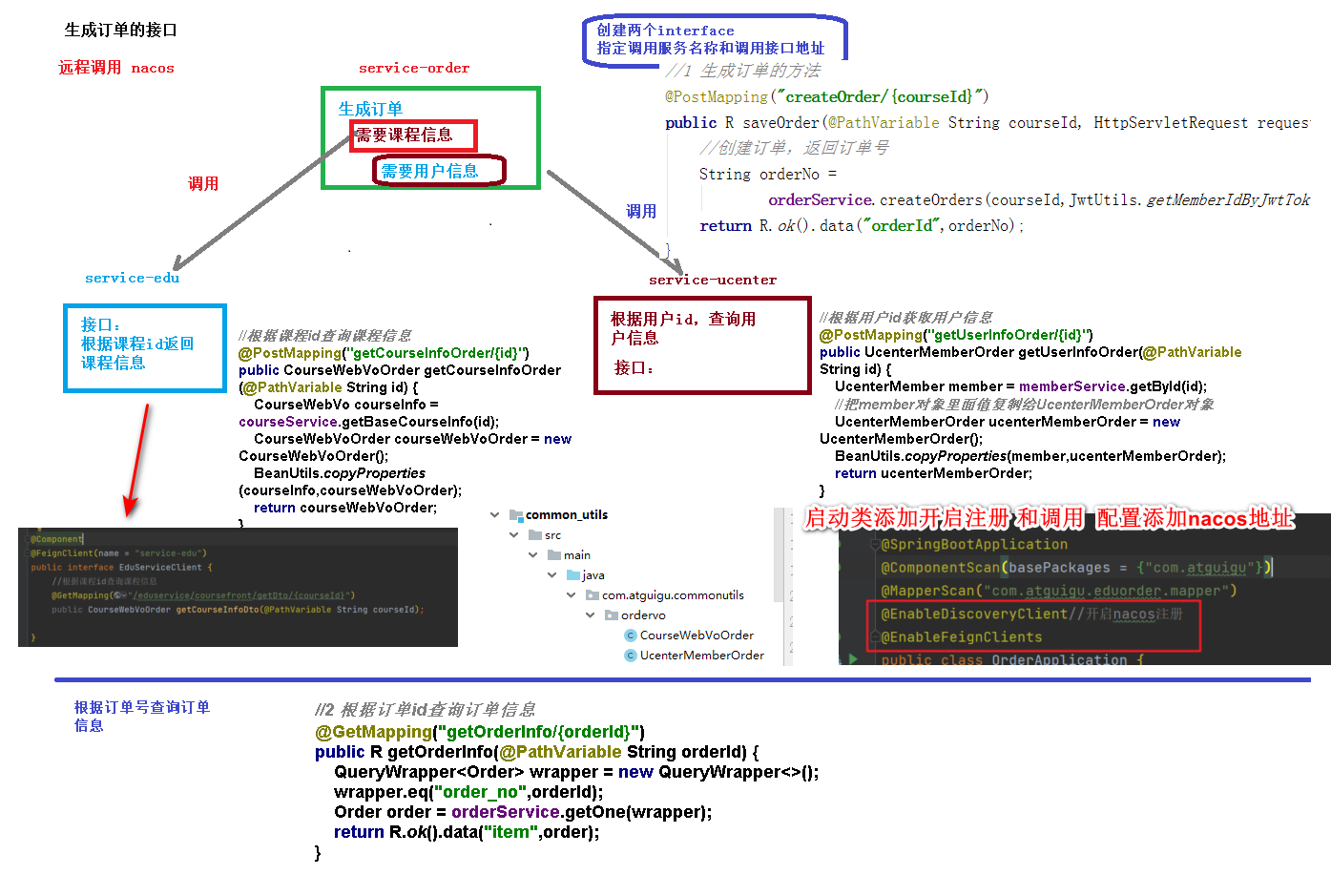


#### （4）使用微信扫描支付后，会跳转回到课程详情页面，同时显示“立即观看”



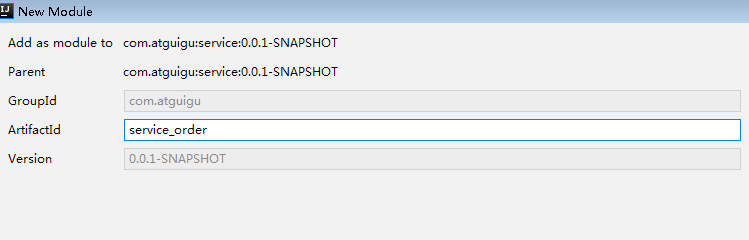
# 02-创建支付模块和开发订单接口





## 一、创建支付模块和准备

### 1、在service模块下创建子模块service\_order



### 2、在service\_order模块中引入依赖

*<*dependencies*>*

*<*dependency*>  
 <*groupId*>*com.github.wxpay*</*groupId*>  
 <*artifactId*>*wxpay-sdk*</*artifactId*>  
 <*version*>*0.0.3*</*version*>  
 </*dependency*>  
 <*dependency*>  
 <*groupId*>*com.alibaba*</*groupId*>  
 <*artifactId*>*fastjson*</*artifactId*>  
 </*dependency*>  
</*dependencies*>*

### 3、创建支付相关的表



#

# Structure for table "t\_order"

#

CREATE TABLE `t\_order` (

  `id` char(19) NOT NULL DEFAULT '',

  `order\_no` varchar(20) NOT NULL DEFAULT '' COMMENT '订单号',

  `course\_id` varchar(19) NOT NULL DEFAULT '' COMMENT '课程id',

  `course\_title` varchar(100) DEFAULT NULL COMMENT '课程名称',

  `course\_cover` varchar(255) DEFAULT NULL COMMENT '课程封面',

  `teacher\_name` varchar(20) DEFAULT NULL COMMENT '讲师名称',

  `member\_id` varchar(19) NOT NULL DEFAULT '' COMMENT '会员id',

  `nickname` varchar(50) DEFAULT NULL COMMENT '会员昵称',

  `mobile` varchar(11) DEFAULT NULL COMMENT '会员手机',

  `total\_fee` decimal(10,2) DEFAULT '0.01' COMMENT '订单金额（分）',

  `pay\_type` tinyint(3) DEFAULT NULL COMMENT '支付类型（1：微信 2：支付宝）',

  `status` tinyint(3) DEFAULT NULL COMMENT '订单状态（0：未支付 1：已支付）',

  `is\_deleted` tinyint(1) unsigned NOT NULL DEFAULT '0' COMMENT '逻辑删除 1（true）已删除， 0（false）未删除',

  `gmt\_create` datetime NOT NULL COMMENT '创建时间',

  `gmt\_modified` datetime NOT NULL COMMENT '更新时间',

  PRIMARY KEY (`id`),

  UNIQUE KEY `ux\_order\_no` (`order\_no`),

  KEY `idx\_course\_id` (`course\_id`),

  KEY `idx\_member\_id` (`member\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='订单';

CREATE TABLE `t\_pay\_log` (

  `id` char(19) NOT NULL DEFAULT '',

  `order\_no` varchar(20) NOT NULL DEFAULT '' COMMENT '订单号',

  `pay\_time` datetime DEFAULT NULL COMMENT '支付完成时间',

  `total\_fee` decimal(10,2) DEFAULT '0.01' COMMENT '支付金额（分）',

  `transaction\_id` varchar(30) DEFAULT NULL COMMENT '交易流水号',

  `trade\_state` char(20) DEFAULT NULL COMMENT '交易状态',

  `pay\_type` tinyint(3) NOT NULL DEFAULT '0' COMMENT '支付类型（1：微信 2：支付宝）',

  `attr` text COMMENT '其他属性',

  `is\_deleted` tinyint(1) unsigned NOT NULL DEFAULT '0' COMMENT '逻辑删除 1（true）已删除， 0（false）未删除',

  `gmt\_create` datetime NOT NULL COMMENT '创建时间',

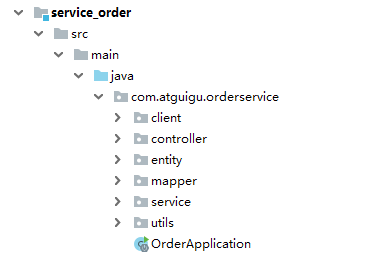
  `gmt\_modified` datetime NOT NULL COMMENT '更新时间',

  PRIMARY KEY (`id`),

  UNIQUE KEY `uk\_order\_no` (`order\_no`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='支付日志表';

### 4、使用代码生成器生成相关代码



public class CodeGenerator *{*

@Test  
 public void run*() {* // 1、创建代码生成器  
 AutoGenerator mpg = new AutoGenerator*()*;  
  
 // 2、全局配置  
 GlobalConfig gc = new GlobalConfig*()*;  
 String projectPath = System.*getProperty(*"user.dir"*)*;  
 //===修改为绝对路径===  
 gc.setOutputDir*(*"D:\\code\\guli\_school\\guli\_parent\\service\\service\_order" + "/src/main/java"*)*;  
 gc.setAuthor*(*"gouwenyong"*)*;  
 gc.setOpen*(*false*)*; //生成后是否打开资源管理器  
 gc.setFileOverride*(*false*)*; //重新生成时文件是否覆盖  
 gc.setServiceName*(*"%sService"*)*; //去掉Service接口的首字母I  
 gc.setIdType*(*IdType.*ID\_WORKER\_STR)*; //主键策略 ID\_WORKER  
 gc.setDateType*(*DateType.*ONLY\_DATE)*;//定义生成的实体类中日期类型  
 gc.setSwagger2*(*true*)*;//开启Swagger2模式  
  
 mpg.setGlobalConfig*(*gc*)*;  
  
 // 3、数据源配置  
 DataSourceConfig dsc = new DataSourceConfig*()*;  
 dsc.setUrl*(*"jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8"*)*;  
 dsc.setDriverName*(*"com.mysql.cj.jdbc.Driver"*)*;  
 dsc.setUsername*(*"root"*)*;  
 dsc.setPassword*(*"root"*)*;  
 dsc.setDbType*(*DbType.*MYSQL)*;  
 mpg.setDataSource*(*dsc*)*;  
  
 // 4、包配置  
 PackageConfig pc = new PackageConfig*()*;  
  
  
 pc.setParent*(*"com.atguigu"*)*;//包名  
 pc.setModuleName*(*"eduorder"*)*; //模块名  
  
 pc.setController*(*"controller"*)*;  
 pc.setEntity*(*"entity"*)*;  
 pc.setService*(*"service"*)*;  
 pc.setMapper*(*"mapper"*)*;  
 mpg.setPackageInfo*(*pc*)*;  
  
 // 5、策略配置  
 StrategyConfig strategy = new StrategyConfig*()*;  
 strategy.setInclude*(* "t\_order","t\_pay\_log"*)*;//对应的表名  
 strategy.setNaming*(*NamingStrategy.*underline\_to\_camel)*;//数据库表映射到实体的命名策略  
 strategy.setTablePrefix*(*"t" + "\_"*)*; //生成实体时去掉表前缀  
  
 strategy.setColumnNaming*(*NamingStrategy.*underline\_to\_camel)*;//数据库表字段映射到实体的命名策略  
 strategy.setEntityLombokModel*(*true*)*; // lombok 模型 @Accessors(chain = true) setter链式操作  
  
 strategy.setRestControllerStyle*(*true*)*; //restful api风格控制器  
 strategy.setControllerMappingHyphenStyle*(*true*)*; //url中驼峰转连字符  
  
 mpg.setStrategy*(*strategy*)*;  
  
  
 // 6、执行  
 mpg.execute*()*;  
 *}  
}*

### 5、编写application.properties配置文件

# 服务端口

server.port=8007  
# 服务名  
spring.application.name=service-order  
# mysql数据库连接  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/guli?serverTimezone=GMT%2B8  
spring.datasource.username=root  
spring.datasource.password=root  
#返回json的全局时间格式  
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss  
spring.jackson.time-zone=GMT+8  
#配置mapper xml文件的路径  
mybatis-plus.mapper-locations=classpath:com/atguigu/orderservice/mapper/xml/\*.xml  
#mybatis日志  
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl  
# nacos服务地址  
spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848  
#开启熔断机制  
feign.hystrix.enabled=true  
# 设置hystrix超时时间，默认1000ms  
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds=3000

## 二、开发创建订单接口

### 1、编写订单controller

@RestController

@RequestMapping("/orderservice/order")  
@CrossOrigin  
public class TOrderController {  
 @Autowired  
 private TOrderService orderService;  
  
 //根据课程id和用户id创建订单，返回订单id  
 @PostMapping("createOrder/{courseId}")  
 public R save(@PathVariable String courseId, HttpServletRequest request) {  
 String orderId = orderService.saveOrder(courseId, JwtUtils.getMemberIdByJwtToken(request));  
 return R.ok().data("orderId", orderId);  
 }  
}

### 2、在service\_edu创建接口

（1）实现根据课程id获取课程信息，返回课程信息对象

//根据课程id查询课程信息

@GetMapping("getDto/{courseId}")  
public com.atguigu.commonutils.vo.CourseInfoForm getCourseInfoDto(@PathVariable String courseId) {  
 CourseInfoForm courseInfoForm = courseService.getCourseInfo(courseId);  
 com.atguigu.commonutils.vo.CourseInfoForm courseInfo = new com.atguigu.commonutils.vo.CourseInfoForm();  
 BeanUtils.copyProperties(courseInfoForm, courseInfo);  
 return courseInfo;  
}

### 3、在service\_ucenter创建接口

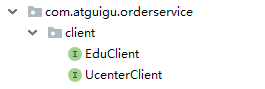
（1）实现用户id获取用户信息，返回用户信息对象

//根据token字符串获取用户信息

@PostMapping("getInfoUc/{id}")  
public com.atguigu.commonutils.vo.UcenterMember getInfo(@PathVariable String id) {  
 //根据用户id获取用户信息  
 UcenterMember ucenterMember = memberService.getById(id);  
 com.atguigu.commonutils.vo.UcenterMember memeber = new com.atguigu.commonutils.vo.UcenterMember();  
 BeanUtils.copyProperties(ucenterMember, memeber);  
 return memeber;  
}

### 4、编写订单service

#### （1）在service\_order模块创建接口，实现远程调用

****

**EduClient**

@Component

@FeignClient("service-edu")  
public interface EduClient {  
 //根据课程id查询课程信息  
 @GetMapping("/eduservice/course/getDto/{courseId}")  
 public com.atguigu.commonutils.vo.CourseInfoForm getCourseInfoDto(@PathVariable("courseId") String courseId);  
}

**UcenterClient**

@Component

@FeignClient("service-ucenter")  
public interface UcenterClient *{* //根据课程id查询课程信息  
 @PostMapping*(*"/ucenterservice/member/getInfoUc/{id}"*)* public com.atguigu.commonutils.vo.UcenterMember getInfo*(*@PathVariable*(*"id"*)* String id*)*;  
*}*

#### （2）在service\_order模块编写创建订单service

@Service

public class TOrderServiceImpl extends ServiceImpl*<*TOrderMapper, TOrder*>* implements TOrderService *{* @Autowired  
 private EduClient eduClient;  
 @Autowired  
 private UcenterClient ucenterClient;  
  
 //创建订单  
 @Override  
 public String saveOrder*(*String courseId, String memberId*) {* //远程调用课程服务，根据课程id获取课程信息  
 CourseInfoForm courseDto = eduClient.getCourseInfoDto*(*courseId*)*;  
  
 //远程调用用户服务，根据用户id获取用户信息  
 UcenterMember ucenterMember = ucenterClient.getInfo*(*memberId*)*;  
 //创建订单  
 TOrder order = new TOrder*()*;  
 order.setOrderNo*(*OrderNoUtil.*getOrderNo())*;  
 order.setCourseId*(*courseId*)*;  
 order.setCourseTitle*(*courseDto.getTitle*())*;  
 order.setCourseCover*(*courseDto.getCover*())*;  
 order.setTeacherName*(*"test"*)*;  
 order.setTotalFee*(*courseDto.getPrice*())*;  
 order.setMemberId*(*memberId*)*;  
 order.setMobile*(*ucenterMember.getMobile*())*;  
 order.setNickname*(*ucenterMember.getNickname*())*;  
 order.setStatus*(*0*)*;  
 order.setPayType*(*1*)*;  
 baseMapper.insert*(*order*)*;  
 return order.getOrderNo*()*;  
 *}  
}*

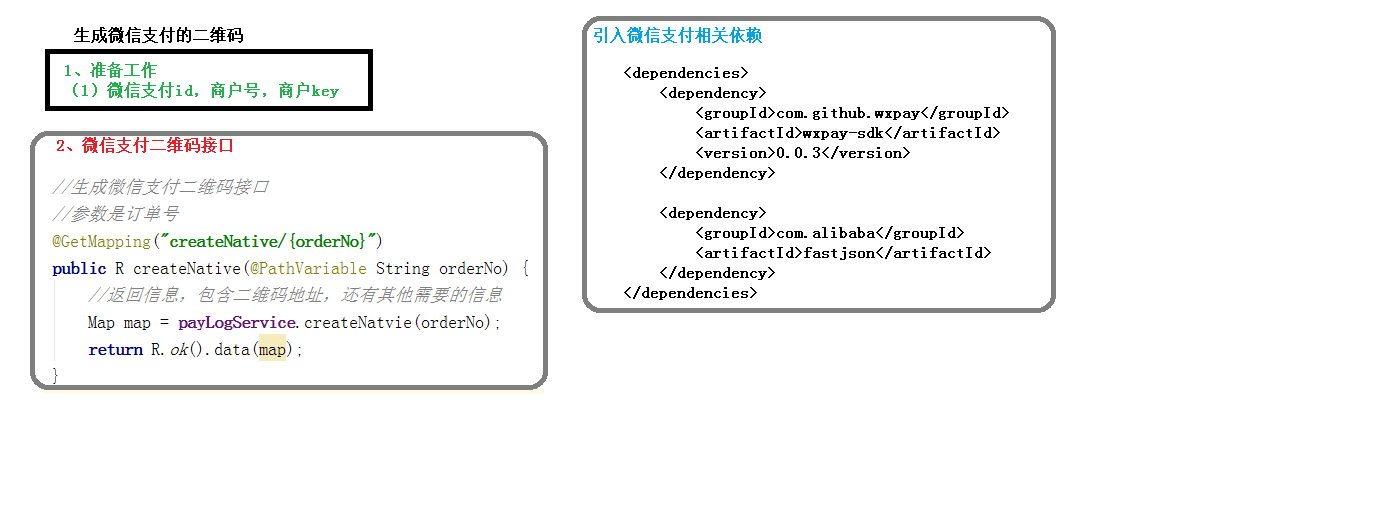
## 三、开发获取订单接口

### 1、在订单controller创建根据id获取订单信息接口

@GetMapping*(*"getOrder/{orderId}"*)*

public R get*(*@PathVariable String orderId*) {* QueryWrapper*<*Order*>* wrapper = new QueryWrapper*<>()*;  
 wrapper.eq*(*"order\_no", orderId*)*;  
 Order order = orderService.getOne*(*wrapper*)*;  
 return R.*ok()*.data*(*"item", order*)*;  
*}*

# 03-开发微信支付接口



## 一、生成微信支付二维码

### 1、编写controller

@RestController

@RequestMapping*(*"/orderservice/log")  
@CrossOrigin  
public class PayLogController {  
 @Autowired  
 private PayLogService payService;  
  
 */\*\*  
 \* 生成二维码  
 \*  
 \** ***@return*** *\*/* @GetMapping("/createNative/{orderNo}")  
 public R createNative(@PathVariable String orderNo) {  
 Map map = payService.createNative(orderNo);  
 return R.ok().data(map);  
 }  
*}*

### 2、编写service

@Service

public class PayLogServiceImpl extends ServiceImpl*<*PayLogMapper, PayLog*>* implements PayLogService *{* @Autowired  
 private TOrderService orderService;  
  
 @Override  
 public Map createNative*(*String orderNo*) {* try *{* //根据订单id获取订单信息  
 QueryWrapper*<*TOrder*>* wrapper = new QueryWrapper*<*>*()*;  
 wrapper.eq*(*"order\_no", orderNo*)*;  
 TOrder order = orderService.getOne*(*wrapper*)*;  
 Map m = new HashMap*()*;  
 //1、设置支付参数  
 m.put*(*"appid", "wx74862e0dfcf69954"*)*;  
 m.put*(*"mch\_id", "1558950191"*)*;  
 m.put*(*"nonce\_str", WXPayUtil.*generateNonceStr())*;  
 m.put*(*"body", order.getCourseTitle*())*;  
 m.put*(*"out\_trade\_no", orderNo*)*;  
 m.put*(*"total\_fee", order.getTotalFee*()*.multiply*(*new BigDecimal*(*"100"*))*.longValue*()* + ""*)*;  
 m.put*(*"spbill\_create\_ip", "127.0.0.1"*)*;  
 m.put*(*"notify\_url", "http://guli.shop/api/order/weixinPay/weixinNotify\n"*)*;  
 m.put*(*"trade\_type", "NATIVE"*)*;  
 //2、HTTPClient来根据URL访问第三方接口并且传递参数  
 HttpClient client = new HttpClient*(*"https://api.mch.weixin.qq.com/pay/unifiedorder"*)*;  
 //client设置参数  
 client.setXmlParam*(*WXPayUtil.*generateSignedXml(*m, "T6m9iK73b0kn9g5v426MKfHQH7X8rKwb"*))*;  
 client.setHttps*(*true*)*;  
 client.post*()*;  
 //3、返回第三方的数据  
 String xml = client.getContent*()*;  
 Map*<*String, String*>* resultMap = WXPayUtil.*xmlToMap(*xml*)*;  
 //4、封装返回结果集  
 Map map = new HashMap*<>()*;  
 map.put*(*"out\_trade\_no", orderNo*)*;  
 map.put*(*"course\_id", order.getCourseId*())*;  
 map.put*(*"total\_fee", order.getTotalFee*())*;  
 map.put*(*"result\_code", resultMap.get*(*"result\_code"*))*;  
 map.put*(*"code\_url", resultMap.get*(*"code\_url"*))*;  
 //微信支付二维码2小时过期，可采取2小时未支付取消订单  
 //redisTemplate.opsForValue().set(orderNo, map, 120, TimeUnit.MINUTES);  
 return map;  
 *}* catch *(*Exception e*) {* e.printStackTrace*()*;  
 return new HashMap*<>()*;  
 *}  
 }  
}*

## 二、获取支付状态接口

### 1、编写controller

@GetMapping*(*"/queryPayStatus/{orderNo}"*)*

public R queryPayStatus*(*@PathVariable String orderNo*) {* //调用查询接口  
 Map*<*String, String*>* map = payService.queryPayStatus*(*orderNo*)*;  
 if *(*map == null*) {*//出错  
 return R.*error()*.message*(*"支付出错"*)*;  
 *}* if *(*map.get*(*"trade\_state"*)*.equals*(*"SUCCESS"*)) {*//如果成功  
 //更改订单状态  
 payService.updateOrderStatus*(*map*)*;  
 return R.*ok()*.message*(*"支付成功"*)*;  
 *}* return R.*ok()*.code*(*25000*)*.message*(*"支付中"*)*;  
*}*

### 2、编写service，更新订单状态

 @Override  
public Map queryPayStatus*(*String orderNo*) {* try *{* //1、封装参数  
 Map m = new HashMap*<>()*;  
 m.put*(*"appid", "wx74862e0dfcf69954"*)*;  
 m.put*(*"mch\_id", "1558950191"*)*;  
 m.put*(*"out\_trade\_no", orderNo*)*;  
 m.put*(*"nonce\_str", WXPayUtil.*generateNonceStr())*;  
 //2、设置请求  
 HttpClient client = new HttpClient*(*"https://api.mch.weixin.qq.com/pay/orderquery"*)*;  
 client.setXmlParam*(*WXPayUtil.*generateSignedXml(*m, "T6m9iK73b0kn9g5v426MKfHQH7X8rKwb"*))*;  
 client.setHttps*(*true*)*;  
 client.post*()*;  
 //3、返回第三方的数据  
 String xml = client.getContent*()*;  
 Map*<*String, String*>* resultMap = WXPayUtil.*xmlToMap(*xml*)*;  
 //6、转成Map  
 //7、返回  
 return resultMap;  
 *}* catch *(*Exception e*) {* e.printStackTrace*()*;  
 *}* return null;  
*}*

@Override

public void updateOrderStatus(Map<String, String> map) {

//获取订单id

String orderNo = map.get("out\_trade\_no");

//根据订单id查询订单信息

QueryWrapper<TOrder> wrapper = new QueryWrapper<>();

wrapper.eq("order\_no",orderNo);

TOrder order = orderService.getOne(wrapper);

if(order.getStatus().intValue() == 1) return;

order.setStatus(1);

orderService.updateById(order);

//记录支付日志

PayLog payLog=new PayLog();

payLog.setOrderNo(order.getOrderNo());//支付订单号

payLog.setPayTime(new Date());

payLog.setPayType(1);//支付类型

payLog.setTotalFee(order.getTotalFee());//总金额(分)

payLog.setTradeState(map.get("trade\_state"));//支付状态

payLog.setTransactionId(map.get("transaction\_id"));

payLog.setAttr(JSONObject.toJSONString(map));

baseMapper.insert(payLog);//插入到支付日志表

}

@Override

public Map queryPayStatus(String orderNo) {

try {

//1、封装参数

Map m = new HashMap<>();

m.put("appid", "wx74862e0dfcf69954");

m.put("mch\_id", "1558950191");

m.put("out\_trade\_no", orderNo);

m.put("nonce\_str", WXPayUtil.generateNonceStr());

//2、设置请求

HttpClient client = new HttpClient("https://api.mch.weixin.qq.com/pay/orderquery");

client.setXmlParam(WXPayUtil.generateSignedXml(m, "T6m9iK73b0kn9g5v426MKfHQH7X8rKwb"));

client.setHttps(true);

client.post();

//3、返回第三方的数据

String xml = client.getContent();

Map<String, String> resultMap = WXPayUtil.xmlToMap(xml);

//6、转成Map

//7、返回

return resultMap;

} catch (Exception e) {

e.printStackTrace();

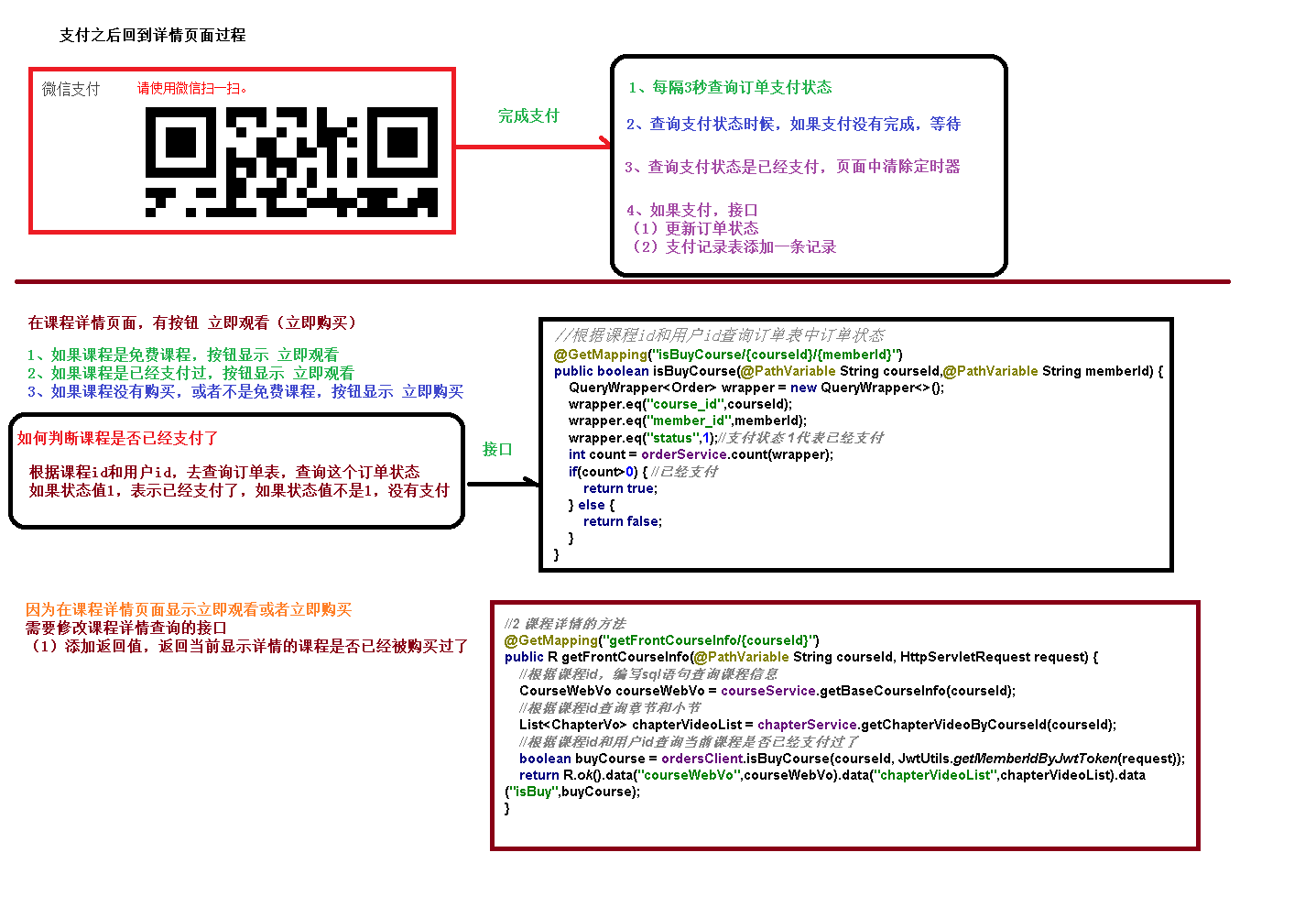
}

return null;

}

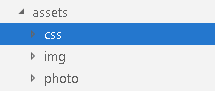
# 04-课程支付前端整合





## 一、页面样式修改

### 1、复制样式文件到assets



### 2、修改default.vue页面

import '~/assets/css/reset.css'

import '~/assets/css/theme.css'

import '~/assets/css/global.css'

import '~/assets/css/web.css'

import '~/assets/css/base.css'

import '~/assets/css/activity\_tab.css'

import '~/assets/css/bottom\_rec.css'

import '~/assets/css/nice\_select.css'

import '~/assets/css/order.css'

import '~/assets/css/swiper-3.3.1.min.css'

import "~/assets/css/pages-weixinpay.css"

## 二、课程支付前端

### 1、在api文件夹下创建order.js文件

import request from '@/utils/request'

export default {

    //1、创建订单

    createOrder(*cid*) {

        return request({

            url: '/orderservice/order/createOrder/'+*cid*,

            method: 'post'

        })

    },

    //2、根据id获取订单

    getById(*cid*) {

        return request({

            url: '/orderservice/order/getOrder/'+*cid*,

            method: 'get'

        })

    },

    //3、生成微信支付二维码

    createNative(*cid*) {

        return request({

            url: '/orderservice/log/createNative/'+*cid*,

            method: 'get'

        })

    },

    //4、根据id获取订单支付状态

    queryPayStatus(*cid*) {

        return request({

            url: '/orderservice/log/queryPayStatus/'+*cid*,

            method: 'get'

        })

    }

}

### 2、在课程详情页面中添加创建订单方法

**在“立即购买”位置添加事件**

**C:\Users\gouwenyong0000\Documents\My Knowledge\temp\58866e87-d5f7-4adf-a034-c6d8d120498b\128\index_files\509e15cd-c1fb-40cf-b047-e89b990f56a8.png**

  methods: {

    //根据课程id，调用接口方法生成订单

    createOrder(){

        order.createOrder(this.courseId).then(*response* *=>* {

            if(*response*.data.success){

                //订单创建成功，跳转到订单页面

                this.$router.push({ path: '/order/'+ *response*.data.data.orderId })

            }

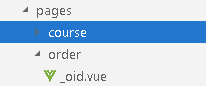
        })

    },

  },

### 3、创建订单页面，显示订单信息

**在pages下面创建order文件夹，创建\_oid.vue页面**

****

**在\_oid.vue页面调用方法，获取订单信息**

#### （1）页面部分

<template>

  <div class="Page Confirm">

    <div class="Title">

      <h1 class="fl f18">订单确认</h1>

      <img src="~/assets/img/cart\_setp2.png" class="fr">

      <div class="clear"></div>

    </div>

    <form name="flowForm" id="flowForm" method="post" action="">

      <table class="GoodList">

        <tbody>

        <tr>

          <th class="name">商品</th>

          <th class="price">原价</th>

          <th class="priceNew">价格</th>

        </tr>

        </tbody>

        <tbody>

        <!-- <tr>

          <td colspan="3" class="Title red f18 fb"><p>限时折扣</p></td>

        </tr> -->

        <tr>

          <td colspan="3" class="teacher">讲师：{{order.teacherName}}</td>

        </tr>

        <tr class="good">

          <td class="name First">

            <a target="\_blank" :href="'https://localhost:3000/course/'+order.courseId">

              <img :src="order.courseCover"></a>

            <div class="goodInfo">

              <input type="hidden" class="ids ids\_14502" value="14502">

              <a target="\_blank" :href="'https://localhost:3000/course/'+ order.courseId">{{order.courseTitle}}</a>

            </div>

          </td>

          <td class="price">

            <p>￥<strong>{{order.totalFee}}</strong></p>

            <!-- <span class="discName red">限时8折</span> -->

          </td>

          <td class="red priceNew Last">￥<strong>{{order.totalFee}}</strong></td>

        </tr>

        <tr>

          <td class="Billing tr" colspan="3">

            <div class="tr">

              <p>共 <strong class="red">1</strong> 件商品，合计<span

                class="red f20">￥<strong>{{order.totalFee}}</strong></span></p>

            </div>

          </td>

        </tr>

        </tbody>

      </table>

      <div class="Finish">

        <div class="fr" id="AgreeDiv">

          <label for="Agree"><p class="on"><input type="checkbox" checked="checked">我已阅读并同意<a href="javascript:" target="\_blank">《谷粒学院购买协议》</a></p></label>

        </div>

        <div class="clear"></div>

        <div class="Main fl">

          <div class="fl">

            <a :href="'/course/'+order.courseId">返回课程详情页</a>

          </div>

          <div class="fr">

            <p>共 <strong class="red">1</strong> 件商品，合计<span class="red f20">￥<strong

              id="AllPrice">{{order.totalFee}}</strong></span></p>

          </div>

        </div>

        <input name="score" value="0" type="hidden" id="usedScore">

        <button class="fr redb" type="button" id="submitPay" @click="toPay()">去支付</button>

        <div class="clear"></div>

      </div>

    </form>

  </div>

</template>

#### （2）调用部分

<script>

import orderApi from '@/api/order'

export default {

  //根据订单id获取订单信息

  asyncData({*params*, *error*}) {

      return orderApi.getById(*params*.oid).then(*response* *=>* {

        return {

            order: *response*.data.data.item

            }

      })

  },

  methods: {

    //点击去支付，跳转到支付页面

    toPay() {

      this.$router.push({path: '/pay/' + this.order.orderNo})

    }

  }

}

</script>

### 4、创建支付页面，生成二维码完成支付

#### （1）页面部分

.\pages\pay\\_oid.vue

<template>

  <div class="cart py-container">

    <!--主内容-->

    <div class="checkout py-container  pay">

      <div class="checkout-tit">

        <h4 class="fl tit-txt"><span class="success-icon"></span><span class="success-info">订单提交成功，请您及时付款！订单号：{{payObj.out\_trade\_no}}</span>

        </h4>

        <span class="fr"><em class="sui-lead">应付金额：</em><em class="orange money">￥{{payObj.total\_fee}}</em></span>

        <div class="clearfix"></div>

      </div>

      <div class="checkout-steps">

        <div class="fl weixin">微信支付</div>

        <div class="fl sao">

          <p class="red">请使用微信扫一扫。</p>

          <div class="fl code">

            <!-- <img id="qrious" src="~/assets/img/erweima.png" alt=""> -->

            <!-- <qriously value="weixin://wxpay/bizpayurl?pr=R7tnDpZ" :size="338"/> -->

            <qriously :value="payObj.code\_url" :size="338"/>

            <div class="saosao">

              <p>请使用微信扫一扫</p>

              <p>扫描二维码支付</p>

            </div>

          </div>

        </div>

        <div class="clearfix"></div>

        <!-- <p><a href="pay.html" target="\_blank">> 其他支付方式</a></p> -->

      </div>

    </div>

  </div>

</template>

#### （2）调用部分

<script>

  import orderApi from '@/api/order

  export default {

    //根据订单id生成微信支付二维码

    asyncData({*params*, *error*}) {

      return orderApi.createNative(*params*.pid).then(*response* *=>* {

        return {

          payObj: *response*.data.data

        }

      })

    },

    data() {

      return {

        timer: null,  // 定时器名称

        initQCode: '',

        timer1:''

      }

    },

    mounted() {

      //在页面渲染之后执行

      //每隔三秒，去查询一次支付状态

      this.timer1 = setInterval(() *=>* {

        this.queryPayStatus(this.payObj.out\_trade\_no)

      }, 3000);

    },

    methods: {

      //查询支付状态的方法

      queryPayStatus(*out\_trade\_no*) {

        orderApi.queryPayStatus(*out\_trade\_no*).then(*response* *=>* {

          if (*response*.data.success) {

              //如果支付成功，清除定时器

              clearInterval(this.timer1)

            this.$message({

              type: 'success',

              message: '支付成功!'

            })

            //跳转到课程详情页面观看视频

            this.$router.push({path: '/course/' + this.payObj.course\_id})

          }

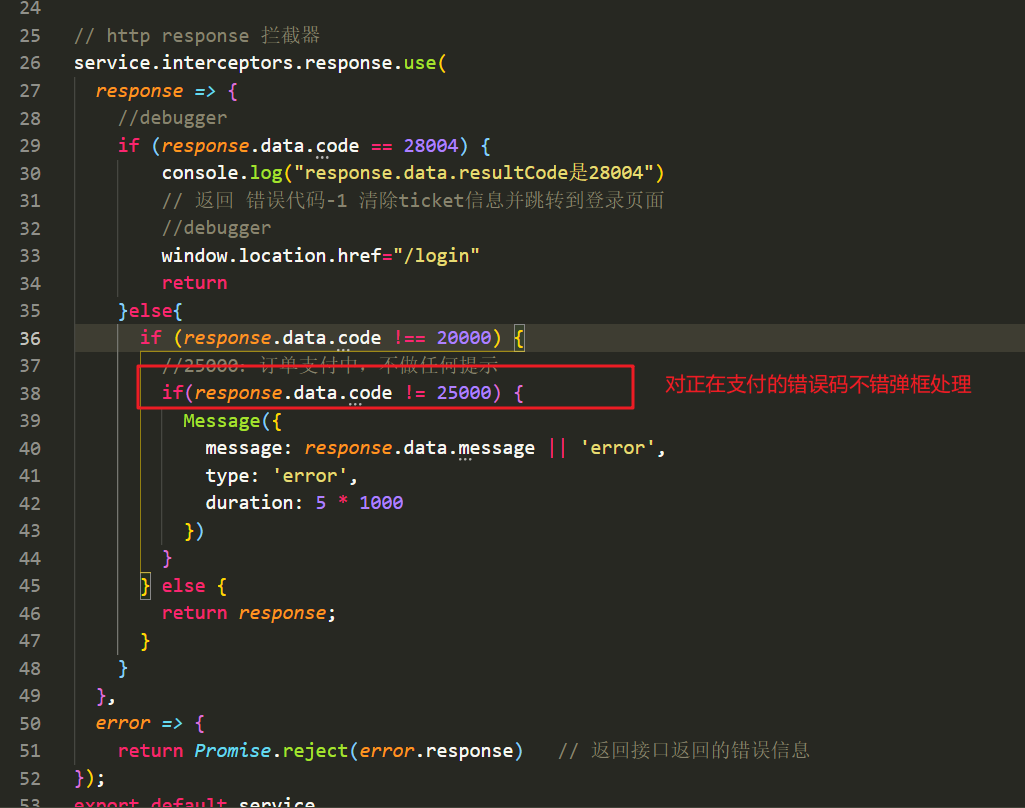
        })

      }

    }

  }

</script>



# 05-课程详情页面功能完善

## 一、修改课程详情接口

### 1、在service\_order模块添加接口

**根据用户id和课程id查询订单信息**

@GetMapping("isBuyCourse/{memberid}/{id}")

public boolean isBuyCourse(@PathVariable String memberid,  
 @PathVariable String id) {  
 //订单状态是1表示支付成功  
 int count = orderService.count(new QueryWrapper<TOrder>().eq("member\_id", memberid).eq("course\_id", id).eq("status", 1));  
 if (count > 0) {  
 return true;  
 } else {  
 return false;  
 }  
}

**2、在service\_edu模块课程详情接口远程调用**

**（1）创建OrderClient接口**

**C:\Users\gouwenyong0000\Documents\My Knowledge\temp\95fa72cf-bb8a-40d7-ab16-ea007e659fe4\128\index_files\5c4b2def-4c36-49b9-8527-f6377a273eb1.png**

@Component

@FeignClient(value = "service-order", fallback = OrderFile.class)  
public interface OrderClient {  
 //查询订单信息  
 @GetMapping("/orderservice/order/isBuyCourse/{memberid}/{id}")  
 public boolean isBuyCourse(@PathVariable("memberid") String memberid, @PathVariable("id") String id);  
}

**（2）在课程详情接口调用**

//根据id查询课程详情信息

@GetMapping("getCourseInfo/{id}")

public R getCourseInfo(@PathVariable String id, HttpServletRequest request) {

//课程查询课程基本信息

CourseFrontInfo courseFrontInfo = courseService.getFrontCourseInfo(id);

//查询课程里面大纲数据

List<ChapterVo> chapterVideoList = chapterService.getChapterVideoById(id);

//远程调用，判断课程是否被购买

boolean buyCourse = orderClient.isBuyCourse(JwtUtils.getMemberIdByJwtToken(request), id);

return R.ok().data("courseFrontInfo",courseFrontInfo).data("chapterVideoList",chapterVideoList).data("isbuy",buyCourse);

}

## 二、修改课程详情页面

### 1、页面内容修改

<section v-if="isbuy || *Number*(course.price)===0" class="c-attr-mt">

              <a href="#" title="立即观看" class="comm-btn c-btn-3" >立即观看</a>

</section>

<section v-else class="c-attr-mt">

    <a href="#" title="立即购买" class="comm-btn c-btn-3" @click="createOrder()">立即购买</a>

</section>

### 2、调用方法修改

<script>

import course from '@/api/course'

export default {

  // asyncData({ params, error }) {//

  //   return course.getCourseInfo(params.id)

  //     .then(response => {

  //       return {

  //         courseInfo: response.data.data.courseFrontInfo,

  //         chapterVideoList: response.data.data.chapterVideoList,

  //         isbuy: response.data.data.isbuy,

  //         courseId:params.id

  //       }

  //     })

  // },

   //和页面异步开始的  如果后台没有获取到header信息，

  asyncData({ *params*, *error* }) {

    return {courseId: *params*.id}

  },

  data() {

    return {

      courseInfo: {},

          chapterVideoList: [],

          isbuy: false,

    }

  },

  created() {

    this.initCourseInfo()

  },

  methods:{

    initCourseInfo() {

      course.getCourseInfo(this.courseId)

        .then(*response* *=>* {

          this.courseInfo=*response*.data.data.courseFrontInfo,

          this.chapterVideoList=*response*.data.data.chapterVideoList,

          this.isbuy=*response*.data.data.isbuy

        })

    },

    createOrder(){

      course.createOrder(this.courseId).then(*response* *=>* {

        if(*response*.data.success){

            this.$router.push({ path: '/order/'+ *response*.data.data.orderId })

        }

      })

    }

  }

};

</script>

### 3、Harder获取不到解决

如果后台获取不到harder信息，不使用aysnData()异步获取数据，采用初始化值，然后created（）里面调用后台获取数据

<script>

import course from "@/api/course";

import order from "@/api/order";

import comment from "@/api/commonedu";

export default {

  //   asyncData({ params, error }) {

  //     return course.getById(params.id).then((response) => {

  //       return {

  //         course: response.data.data.course,

  //         chapterList: response.data.data.chapterVoList,

  //         courseId: params.id,

  //         isbuy: response.data.data.isbuy,

  //       };

  //     });

  //   },

  data() {

    return {

      data: [],

      page: 1,

      limit: 4,

      total: 10,

      comment: {

        content: "",

        courseId: "",

      },"isbuy": false,

    "chapterVoList": [

      {

        "id": "",

        "title": "",

        "children": [

          {

            "id": "",

            "title": "",

            "free": null,

            "videoSourceId": ""

          }

        ]

      }

    ],

    "course": {

      "id": "",

      "title": "",

      "price": 0,

      "lessonNum": 0,

      "cover": "",

      "buyCount": 0,

      "viewCount": 0,

      "description": null,

      "teacherId": "",

      "teacherName": "",

      "intro": "",

      "avatar": "",

      "subjectLevelOneId": "",

      "subjectLevelOne": "",

      "subjectLevelTwoId": "",

      "subjectLevelTwo": ""

    }

    };

  },

  created() {

    this.initCourseInfo();

    this.initComment();

  },

  methods: {

    //查询课程详情信息

    initCourseInfo() {

        debugger

      course.getById(this.$route.params.id).then((*response*) *=>* {

          debugger

        this.course = *response*.data.data.course;

        this.chapterList = *response*.data.data.chapterVoList;

        this.courseId = this.$route.params.id;

        this.isbuy = *response*.data.data.isbuy;

      });

    },

    initComment() {

      comment

        .getPageList(this.page, this.limit, this.courseId)

        .then((*response*) *=>* {

          this.data = *response*.data.data;

        });

    },

    addComment() {

      this.comment.courseId = this.courseId;

      this.comment.teacherId = this.course.teacherId;

      comment.addComment(this.comment).then((*response*) *=>* {

        if (*response*.data.success) {

          this.comment.content = "";

          this.initComment();

        }

      });

    },

    gotoPage(*page*) {

      comment.getPageList(*page*, this.limit, this.courseId).then((*response*) *=>* {

        this.data = *response*.data.data;

      });

    },

    //根据课程id，调用接口方法生成订单

    createOrder() {

      order.createOrder(this.courseId).then((*response*) *=>* {

        if (*response*.data.success) {

          //订单创建成功，跳转到订单页面

          this.$router.push({

            path: "/order/" + *response*.data.data.orderId,

          });

        }

      });

    },

  },

};

</script>