



品优购电商系统开发

第 18 章

微信扫码支付

传智播客.黑马程序员

课程目标

- 目标 1：掌握二维码生成插件 qrious 的使用
- 目标 2：能够说出微信支付开发的整体思路
- 目标 3：能够调用微信支付接口（统一下单）生成支付二维码
- 目标 4：能够调用微信支付接口（查询订单）查询支付状态
- 目标 5：实现支付日志的生成与订单状态的修改

1.二维码

1.1 什么是二维码

二维码又称 QR Code，QR 全称 Quick Response，是一个近几年来移动设备上超流行的一种编码方式，它比传统的 Bar Code 条形码能存更多的信息，也能表示更多的数据类型。

二维条码/二维码（2-dimensional bar code）是用某种特定的几何图形按一定规律在平面（二维方向上）分布的黑白相间的图形记录数据符号信息的；在代码编制上巧妙地利用构成计算机内部逻辑基础的“0”、“1”比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图象输入设备或光电扫描设备自动识读以实现信息自动处理：它具有条码技术的一些共性：每种码制有其特定的字符集；每个字符占有一定的宽度；具有一定的校验功能等。同时还具有对不同行的信息自动识别功能、及处理图形旋转变换点。





1.2 二维码优势

- 信息容量大, 可以容纳多达 1850 个大写字母或 2710 个数字或 500 多个汉字
- 应用范围广, 支持文字,声音,图片,指纹等等...
- 容错能力强, 即使图片出现部分破损也能使用
- 成本低, 容易制作

1.3 二维码容错级别

L 级（低） 7%的码字可以被恢复。

M 级（中） 的码字的 15%可以被恢复。

Q 级（四分）的码字的 25%可以被恢复。

H 级（高） 的码字的 30%可以被恢复。

1.4 二维码生成插件 qrious

qrrious 是一款基于 HTML5 Canvas 的纯 JS 二维码生成插件。通过 qrrious.js 可以快速生成各种二维码，你可以控制二维码的尺寸颜色，还可以将生成的二维码进行 Base64 编码。

qrrious.js 二维码插件的可用配置参数如下：

参数	类型	默认值	描述
background	String	"white"	二维码的背景颜色。
foreground	String	"black"	二维码的前景颜色。
level	String	"L"	二维码的误差校正级别(L, M, Q, H)。
mime	String	"image/png"	二维码输出为图片时的 MIME 类型。
size	Number	100	二维码的尺寸，单位像素。
value	String	""	需要编码为二维码的值

下面的代码即可生成一张二维码



```
<html>

<head>

<title>二维码入门小 demo</title>

</head>

<body>

<img id="qrrious">

<script src="qrrious.min.js"></script>

<script>

    var qr = new QRrious({

        element:document.getElementById('qrrious'),

        size:250,    level:'H',    value:'http://www.itcast.cn'

    });

</script>

</body>

</html>
```

运行效果：



大家掏出手机，扫一下看看是否会看到传智播客的官网呢？



2.微信扫码支付简介

2.1 微信扫码支付申请

微信扫码支付是商户系统按微信支付协议生成支付二维码，用户再用微信“扫一扫”完成支付的模式。该模式适用于 PC 网站支付、实体店单品或订单支付、媒体广告支付等场景。

申请步骤：（了解）

第一步：注册公众号（类型须为：**服务号**）

请根据营业执照类型选择以下主体注册：[个体工商户](#) | [企业/公司](#) | [政府](#) | [媒体](#) | [其他类型](#)。

第二步：认证公众号

公众号认证后才可申请微信支付，认证费：300 元/次。

第三步：提交资料申请微信支付

登录公众平台，点击左侧菜单【微信支付】，开始填写资料等待审核，审核时间为 1-5 个工作日内。

第四步：开户成功，登录商户平台进行验证

资料审核通过后，请登录联系人邮箱查收商户号和密码，并登录商户平台填写财付通备付金打的小额资金数额，完成账户验证。

第五步：在线签署协议

本协议为线上电子协议，签署后方可进行交易及资金结算，签署完立即生效。

本课程已经提供好“传智播客”的微信支付账号，学员无需申请。

2.2 开发文档

微信支付接口调用的整体思路：

按 API 要求组装参数，以 XML 方式发送（POST）给微信支付接口（URL），微信支付接口也是以 XML 方式给予响应。程序根据返回的结果（其中包括支付 URL）生成二维码或判



断订单状态。

在线微信支付开发文档：

<https://pay.weixin.qq.com/wiki/doc/api/index.html>

如果你不能联网，请查阅讲义配套资源（资源\配套软件\微信扫码支付\开发文档）

我们在本章课程中会用到“统一下单”和“查询订单”两组 API

1. appid: 微信公众账号或开放平台 APP 的唯一标识
2. mch_id: 商户号（配置文件中的 partner）
3. partnerkey: 商户密钥
4. sign: 数字签名，根据微信官方提供的密钥和一套算法生成的一个加密信息，就是为了保证交易的安全性

2.3 微信支付 SDK

微信支付提供了 SDK，大家下载后打开源码，install 到本地仓库。

SDK与DEMO下载

平台和语言	说明	支付模式	操作
JAVA	【微信支付】API对应的SDK和调用示例	刷卡支付、公众号支付、扫码支付	下载
.NET C#	【微信支付】API对应的SDK和调用示例	刷卡支付、公众号支付、扫码支付	下载
PHP	【微信支付】API对应的SDK和调用示例	刷卡支付、公众号支付、扫码支付	下载

课程配套的本地仓库已经提供 jar 包，所以安装 SDK 步骤省略。

使用微信支付 SDK,在 maven 工程中引入依赖

```
<dependency>

    <groupId>com.github.wxpay</groupId>

    <artifactId>wxpay-sdk</artifactId>

    <version>0.0.3</version>

</dependency>
```



我们主要会用到微信支付 SDK 的以下功能：

(1) 获取随机字符串

```
WXPAYUtil.generateNonceStr()
```

(2) MAP 转换为 XML 字符串（自动添加签名）

```
WXPAYUtil.generateSignedXml(param, partnerkey)
```

(3) XML 字符串转换为 MAP

```
WXPAYUtil.xmlToMap(result)
```

2.4 HttpClient 工具类

HttpClient 是 Apache Jakarta Common 下的子项目，用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。HttpClient 已经应用在很多的项目中，比如 Apache Jakarta 上很著名的另外两个开源项目 Cactus 和 HTMLUnit 都使用了 HttpClient。

HttpClient 通俗的讲就是模拟了浏览器的行为，如果我们需要在后端向某一地址提交数据获取结果，就可以使用 HttpClient。

关于 HttpClient（原生）具体的使用不属于我们本章的学习内容，我们这里这里为了简化 HttpClient 的使用，提供了工具类 HttpClient（对原生 HttpClient 进行了封装）

HttpClient 工具类使用的步骤

```
HttpClient client=new HttpClient(请求的url地址);

client.setHttps(true); //是否是https协议

client.setXmlParam(xmlParam); //发送的xml数据

client.post(); //执行post请求

String result = client.getContent(); //获取结果
```



2.5 工程搭建与准备工作

(1) 建立支付服务接口模块 pinyougou-pay-interface (jar)

(2) 建立支付服务实现模块 pinyougou-pay-service (war) 依赖 pinyougou-pay-interface 和 pinyougou-common 、 spring dubbox 相关依赖 、 微信 SDK (因为不需要连接数据库所以不用引用 dao 工程)

```
<dependency>

    <groupId>com.github.wxpay</groupId>

    <artifactId>wxpay-sdk</artifactId>

    <version>0.0.3</version>

</dependency>
```

添加 tomcat 插件，运行端口为 9000

添加 spring 配置文件，参见其它服务工程

(3) 在 pinyougou-common 工程中添加工具类 HttpClient.java，并添加依赖

```
<dependency>

    <groupId>org.apache.httpcomponents</groupId>

    <artifactId>httpclient</artifactId>

</dependency>
```

添加配置文件 weixinpay.properties

```
appid=wx8397f8696b538317

partner=1473426802

partnerkey=8A627A4578ACE384017C997F12D68B23

notifyurl=http://a31ef7db.ngrok.io/WeChatPay/WeChatPayNotify
```

appid: 微信公众账号或开放平台 APP 的唯一标识

partner: 财付通平台的商户账号

partnerkey: 财付通平台的商户密钥

notifyurl: 回调地址

(4) pinyougou-cart-web 依赖工程 pinyougou-pay-service

(5) 将二维码插件 QRious 拷贝到 pinyougou-cart-web 的 plugins 目录中

3.品优购-微信支付二维码生成

3.1 需求分析与实现思路

3.1.1 需求分析

在支付页面上生成支付二维码，并显示订单号和金额

用户拿出手机,打开微信扫描页面上的二维码,然后在微信中完成支付



✓ 订单提交成功，请您及时付款！订单号：201701010001

应付金额：¥0.01元



3.1.2 实现思路

我们通过 HttpClient 工具类实现对远程支付接口的调用。

接口链接：<https://api.mch.weixin.qq.com/pay/unifiedorder>

具体参数参见“统一下单”API，构建参数发送给统一下单的 url，返回的信息中有支付 url，



根据 url 生成二维码，显示的订单号和金额也在返回的信息中。

3.2 后端代码实现

3.2.1 服务接口层

(1) 在 pinyougou-pay-interface 创建包 com.pinyougou.pay.service，包下建立接口

```
package com.pinyougou.pay.service;

import java.util.Map;

/**
 * 微信支付接口
 *
 * @author Administrator
 *
 */

public interface WeixinPayService {

    /**
     * 生成微信支付二维码
     *
     * @param out_trade_no 订单号
     * @param total_fee 金额(分)
     * @return
     */

    public Map createNative(String out_trade_no,String total_fee);

}
```



3.2.2 服务实现层

pinyougou-pay-service 创建 com.pinyougou.pay.service.impl 包，新建类

```
@Service

public class WeixinPayServiceImpl implements WeixinPayService {

    @Value("${appid}")

    private String appid;

    @Value("${partner}")

    private String partner;

    @Value("${partnerkey}")

    private String partnerkey;

    /**
     * 生成二维码
     * @return
     */

    public Map createNative(String out_trade_no,String total_fee){

        //1.创建参数

        Map<String,String> param=new HashMap();//创建参数

        param.put("appid", appid);//公众号

        param.put("mch_id", partner);//商户号
```



```
param.put("nonce_str", WXPUtil.generateNonceStr()); //随机字符串

param.put("body", "品优购"); //商品描述

param.put("out_trade_no", out_trade_no); //商户订单号

param.put("total_fee", total_fee); //总金额（分）

param.put("spbill_create_ip", "127.0.0.1"); //IP

param.put("notify_url", "http://test.itcast.cn"); //回调地址(随便写)

param.put("trade_type", "NATIVE"); //交易类型

try {

    //2.生成要发送的 xml

    String xmlParam = WXPUtil.generateSignedXml(param, partnerkey);

    System.out.println(xmlParam);

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

    client.setHttps(true);

    client.setXmlParam(xmlParam);

    client.post();

    //3.获得结果

    String result = client.getContent();

    System.out.println(result);

    Map<String, String> resultMap = WXPUtil.xmlToMap(result);

    Map<String, String> map = new HashMap<>();

    map.put("code_url", resultMap.get("code_url")); //支付地址

    map.put("total_fee", total_fee); //总金额

    map.put("out_trade_no", out_trade_no); //订单号
```



```
        return map;

    } catch (Exception e) {

        e.printStackTrace();

        return new HashMap<>();

    }

}

}
```

3.2.3 控制层

pinyougou-cart-web 创建 PayController.java

```
/**

 * 支付控制层

 * @author Administrator

 *

 */

@RestController

@RequestMapping("/pay")

public class PayController {

    @Reference

    private WeixinPayService weixinPayService;

    /**

     * 生成二维码
```



```
* @return

*/

@RequestMapping("/createNative")

public Map createNative(){

    IdWorker idworker=new IdWorker();

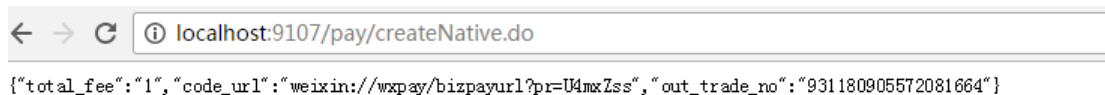
    return weixinPayService.createNative(idworker.nextId()+"","1");

}

}
```

这里我们订单号通过分布式 ID 生成器生成，金额暂时写死，后续开发我们再对接业务系统得到订单号和金额

浏览器测试



```
{"total_fee":"1","code_url":"weixin://wxpay/bizpayurl?pr=U4mxZss","out_trade_no":"931180905572081664"}
```

3.3 前端代码实现

3.3.1 服务层

在 pinyougou-cart-web 创建 payService.js

```
app.service('payService',function($http){

    //本地支付

    this.createNative=function(){

        return $http.get('pay/createNative.do');

    }

});
```



3.3.2 控制层

在 pinyougou-cart-web 创建 payController.js

```
app.controller('payController' ,function($scope ,payService){

    //本地生成二维码

    $scope.createNative=function(){

        payService.createNative().success(

            function(response){

                $scope.money= (response.total_fee/100).toFixed(2) ;    //金额

                $scope.out_trade_no= response.out_trade_no;//订单号

                //二维码

                var qr = new QRious({

                    element:document.getElementById('qrious'),

                    size:250,

                    level:'H',

                    value:response.code_url

                });

            }

        );

    }

});
```

3.3.3 页面

修改 pay.html ,引入 js



```
<script type="text/javascript" src="plugins/angularjs/angular.min.js"> </script>

<script type="text/javascript" src="js/base.js"> </script>

<script type="text/javascript" src="js/service/payService.js"> </script>

<script type="text/javascript" src="js/controller/payController.js"> </script>

<script type="text/javascript" src="plugins/qrious.min.js"></script>
```

指令

```
<body ng-app="pinyougou" ng-controller="payController" ng-init="createNative()">
```

设置二维码图片的 ID

```
<p class="red"></p>

    <div class="fl code">

        <img id="qrrious">

        <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>
```

显示订单号

```
订单号: {{out_trade_no}}
```

显示金额

```
<em class="orange money">¥{{money}}</em>元
```




4.品优购-检测支付状态

4.1 需求分析及实现思路

4.1.1 需求分析

当用户支付成功后跳转到成功页面



恭喜您，支付成功啦！

支付方式：微信支付

支付金额：¥1006.00元

查看订单

继续购物

当返回异常时跳转到错误页面



支付失败，请稍后再试

失败原因：不能使用金币购买！

您可以先去 [品优购首页](#) 逛逛

重新支付

4.1.2 实现思路

我们通过 HttpClient 工具类实现对远程支付接口的调用。

接口链接：<https://api.mch.weixin.qq.com/pay/orderquery>

具体参数参见“查询订单”API，我们在 controller 方法中轮询调用查询订单（间隔 3 秒），当返回状态为 success 时，我们会在 controller 方法返回结果。前端代码收到结果后跳转到成功页面。



4.2 检测支付状态-后端代码

4.2.1 服务接口层

在 pinyougou-pay-interface 的 WeixinPayService.java 中新增方法定义

```
/**
 * 查询支付状态
 * @param out_trade_no
 */
public Map queryPayStatus(String out_trade_no);
```

4.2.2 服务实现层

在 pinyougou-pay-service 的 WeixinPayServiceImpl.java 中实现方法

```
@Override
public Map queryPayStatus(String out_trade_no) {
    Map param=new HashMap();

    param.put("appid", appid);//公众账号 ID
    param.put("mch_id", partner);//商户号
    param.put("out_trade_no", out_trade_no);//订单号
    param.put("nonce_str", WXPUtil.generateNonceStr());//随机字符串

    String url="https://api.mch.weixin.qq.com/pay/orderquery";

    try {
        String xmlParam = WXPUtil.generateSignedXml(param, partnerkey);

        HttpClient client=new HttpClient(url);

        client.setHttps(true);
```



```
        client.setXmlParam(xmlParam);

        client.post();

        String result = client.getContent();

        Map<String, String> map = WXPAYUtil.xmlToMap(result);

        System.out.println(map);

        return map;

    } catch (Exception e) {

        e.printStackTrace();

        return null;

    }

}
```

4.2.3 控制层

在 pinyougou-cart-web 的 PayController.java 新增方法

```
/**

 * 查询支付状态

 * @param out_trade_no

 * @return

 */

@RequestMapping("/queryPayStatus")

public Result queryPayStatus(String out_trade_no){

    Result result=null;

    while(true){
```



```
//调用查询接口

Map<String,String> map = weixinPayService.queryPayStatus\(out\_trade\_no\);

if(map==null){//出错

    result=new Result(false, "支付出错");

    break;

}

if(map.get("trade_state").equals("SUCCESS")){//如果成功

    result=new Result(true, "支付成功");

    break;

}

try {

    Thread.sleep(3000);//间隔三秒

} catch (InterruptedException e) {

    e.printStackTrace();

}

}

return result;

}
```

4.3 检测支付状态-前端代码

4.3.1 服务层

在 payService.js 新增方法

```
//查询支付状态
```



```
this.queryPayStatus=function(out_trade_no){  
  
    return $http.get('pay/queryPayStatus.do?out_trade_no='+out_trade_no);  
  
}
```

4.3.2 控制层

在 payController.js 中新增方法

```
//查询支付状态  
  
queryPayStatus=function(out_trade_no){  
  
    payService.queryPayStatus(out_trade_no).success(  
  
        function(response){  
  
            if(response.success){  
  
                location.href="paysuccess.html";  
  
            }else{  
  
                location.href="payfail.html";  
  
            }  
  
        }  
  
    );  
  
}
```

在 createNative 方法的回调方法中调用此查询方法

```
//本地生成二维码  
  
$scope.createNative=function(){  
  
    payService.createNative().success(  
  
        function(response){
```



```
.....

        queryPayStatus(response.out_trade_no);//查询支付状态

    }

};

}
```

4.4 查询时间限制

4.4.1 问题分析

如果用户到了二维码页面一直未支付，或是关掉了支付页面，我们的代码会一直循环调用微信接口，这样会对程序造成很大的压力。所以我们要加一个时间限制或是循环次数限制，当超过时间或次数时，跳出循环。

4.4.2 代码完善

(1) 修改 pinyougou-cart-web 工程 PayController.java 的 queryPayStatus 方法

```
@RequestMapping("/queryPayStatus")

public Result queryPayStatus(String out_trade_no){

    Result result=null;

    int x=0;

    while(true){

        //调用查询接口

        .....

        try {

            Thread.sleep(3000);//间隔三秒

        } catch (InterruptedException e) {
```



```
        e.printStackTrace();

    }

    //为了不让循环无休止地运行，我们定义一个循环变量，如果这个变量超过了这个值则退出循环，设置时间为 5 分钟

    x++;

    if(x>=100){

        result=new Result(false, "二维码超时");

        break;

    }

}

return result;

}
```

(2) 修改 payController.js

```
//查询支付状态

queryPayStatus=function(out_trade_no){

    payService.queryPayStatus(out_trade_no).success(

        function(response){

            if(response.success){

                location.href="paysuccess.html";

            }else{

                if(response.message=='二维码超时'){

                    $scope.createNative();//重新生成二维码

                }else{


```



```
        location.href="payfail.html";
    }
}
}
);
}
```

4.5 支付成功页面显示金额

4.5.1 问题分析

现在我们支付成功页面显示的是固定的值，怎么显示真正的支付金额呢？我们这里可以使用 angularJS 的页面传参来解决。

4.5.2 代码完善

(1) 修改 payController.js 跳转页面传参

```
//查询支付状态

queryPayStatus=function(out_trade_no){

    payService.queryPayStatus(out_trade_no).success(

        function(response){

            if(response.success){

                location.href="paysuccess.html?money="+$scope.money;

            }else{

                if(response.message=='二维码超时'){

                    $scope.createNative();//重新生成二维码

                }else{

                }

            }

        }

    );

}
```




```
        location.href="payfail.html";

    }

}

}

);

}
```

(2) 在 payController.js 中引入 \$location 服务，新增方法

```
//获取金额

$scope.getMoney=function(){

    return $location.search()['money'];

}
```

(3) 修改页面 paysuccess.html，引入 JS (与 pay.html 相同)，body 添加指令

```
ng-app="pinyougou" ng-controller="payController"
```

用表达式显示金额

```
<p>支付金额: ¥{{getMoney()}}元</p>
```

5. 品优购-支付日志

5.1 需求分析

我们现在系统还有两个问题需要解决：

- (1) 系统中无法查询到支付记录
- (2) 支付后订单状态没有改变

我们现在就来解决这两个问题。

实现思路：



(1) 在用户下订单时，判断如果为微信支付，就想支付日志表添加一条记录，信息包括支付总金额、订单 ID（多个）、用户 ID、下单时间等信息，支付状态为 0（未支付）

(2) 生成的支付日志对象放入 redis 中，以用户 ID 作为 key，这样在生成支付二维码时就可以从 redis 中提取支付日志对象中的金额和订单号。

(3) 当用户支付成功后，修改支付日志的支付状态为 1（已支付），并记录微信传递给我们的交易流水号。根据订单 ID（多个）修改订单的状态为 2（已付款）。

5.2 表结构分析

tb_paylog 支付日志表

字段	类型	长度	含义
out_trade_no	varchar	30	支付订单号
create_time	datetime		创建时间
pay_time	datetime		支付完成时间
total_fee	bigint		支付金额（分）
transaction_id	varchar	30	交易流水号
trade_state	varchar	1	交易状态
pay_type	varchar	1	支付类型： 1:微信 2:支付宝 3:网银
order_list	varchar	200	订单表 ID 串，用逗号分隔

5.3 插入日志记录

修改 pinyougou-order-service 工程 OrderServiceImpl.java 的 add 方法。

内容：判断如果支付方式为微信支付，向数据库插入支付日志记录，并放入 redis 存储



```
@Autowired

private TbPayLogMapper payLogMapper;

/**
 * 增加
 */

public void add(TbOrder order) {

    List<Cart> cartList = (List<Cart>)

        redisTemplate.boundHashOps("cartList").get( order.getUserId() );

    List<String> orderIdList=new ArrayList();//订单 ID 列表

    double total_money=0;//总金额 （元）

    for(Cart cart:cartList){

        long orderId = idWorker.nextId();

        .....

        orderIdList.add(orderId+"");//添加到订单列表

        total_money+=money;//累加到总金额

    }

    if("1".equals(order.getPaymentType())){//如果是微信支付

        TbPayLog payLog=new TbPayLog();

        String outTradeNo= idWorker.nextId()+"";//支付订单号

        payLog.setOutTradeNo(outTradeNo);//支付订单号

        payLog.setCreateTime(new Date());//创建时间

        //订单号列表，逗号分隔

        String ids=orderIdList.toString().replace("[", "").replace("]",
        "").replace(" ", "");
    }
```



```
        payLog.setOrderList(ids);//订单号列表，逗号分隔

        payLog.setPayType("1");//支付类型

        payLog.setTotalFee( (long)(total_money*100 ) );//总金额(分)

        payLog.setTradeState("0");//支付状态

        payLog.setUserId(order.getUserId());//用户 ID

        payLogMapper.insert(payLog);//插入到支付日志表

        redisTemplate.boundHashOps("payLog").put(order.getUserId(), payLog);//
放入缓存

    }

    redisTemplate.boundHashOps("cartList").delete(order.getUserId());

}
```

5.4 读取支付日志

5.4.1 服务接口层

pinyougou-order-interface 工程的 OrderService.java 新增方法

```
/**

 * 根据用户查询 payLog

 * @param userId

 * @return

 */

public TbPayLog searchPayLogFromRedis(String userId);
```

5.4.2 服务实现层

pinyougou-order-service 的 OrderServiceImpl.java 实现方法



```
@Override

public TbPayLog searchPayLogFromRedis(String userId) {

    return (TbPayLog) redisTemplate.boundHashOps("payLog").get(userId);

}
```

5.4.3 控制层

修改 pinyougou-cart-web 工程 PayController.java 的 createNative 方法

实现思路：调用获取支付日志对象的方法，得到订单号和金额

```
@Reference

private OrderService orderService;

/**
 * 生成二维码
 * @return
 */

@RequestMapping("/createNative")

public Map createNative(){

    //获取当前用户

    String
userId=SecurityContextHolder.getContext().getAuthentication().getName();

    //到 redis 查询支付日志

    TbPayLog payLog = orderService.searchPayLogFromRedis(userId);

    //判断支付日志存在

    if(payLog!=null){

        return
```



```
weixinPayService.createNative(payLog.getOutTradeNo(),payLog.getTotalFee()+"" );

        }else{

            return new HashMap();

        }

    }

}
```

5.5 修改订单状态

5.5.1 服务接口层

在 pinyougou-order-interface 的 OrderService.java 新增方法定义

```
/**

 * 修改订单状态

 * @param out_trade_no 支付订单号

 * @param transaction_id 微信返回的交易流水号

 */

public void updateOrderStatus(String out_trade_no,String transaction_id);
```

5.5.2 服务实现层

在 pinyougou-order-service 工程 OrderServiceImpl.java 实现该方法.

这个方法主要做三件事：

1. 修改支付日志状态
2. 修改关联的订单的状态
3. 清除缓存中的支付日志对象

```
@Override
```



```
public void updateOrderStatus(String out_trade_no, String transaction_id) {

    //1.修改支付日志状态

    TbPayLog payLog = payLogMapper.selectByPrimaryKey(out_trade_no);

    payLog.setPayTime(new Date());

    payLog.setTradeState("1");//已支付

    payLog.setTransactionId(transaction_id);//交易号

    payLogMapper.updateByPrimaryKey(payLog);

    //2.修改订单状态

    String orderList = payLog.getOrderList();//获取订单号列表

    String[] orderIds = orderList.split(",");//获取订单号数组

    for(String orderId:orderIds){

        TbOrder order =
orderMapper.selectByPrimaryKey( Long.parseLong(orderId) );

        if(order!=null){

            order.setStatus("2");//已付款

            orderMapper.updateByPrimaryKey(order);

        }

    }

    //清除 redis 缓存数据

    redisTemplate.boundHashOps("payLog").delete(payLog.getUserId());

}
```



5.5.3 控制层

修改 pinyougou-cart-web 的 PayController.java。在微信支付接口有成功返回状态时，调用修改状态的方法

```
/**
 * 查询支付状态
 *
 * @param out_trade_no
 *
 * @return
 */
@RequestMapping("/queryPayStatus")
public Result queryPayStatus(String out_trade_no){

    Result result=null;

    int x=0;

    while(true){

        //调用查询接口

        Map<String,String> map = weixinPayService.queryPayStatus(out_trade_no);

        if(map==null){//出错

            result=new Result(false, "支付出错");

            break;

        }

        if(map.get("trade_state").equals("SUCCESS")){//如果成功

            result=new Result(true, "支付成功");

            //修改订单状态

            orderService.updateOrderStatus(out_trade_no,
map.get("transaction_id"));
```




```
        break;

    }

    try {

        Thread.sleep(3000); //间隔三秒

    } catch (InterruptedException e) {

        e.printStackTrace();

    }

    //为了不让循环无休止地运行，我们定义一个循环变量，如果这个变量超过了这个值则退出循环，设置时间为 5 分钟

    .....

}

return result;

}
```

5.6 支付日志显示（学员实现）

需求：在运营商后台中，显示支付日志列表，实现按日期、状态、用户进行查询。

学员实现。