尚医通

版本：V1.0

# 一、微信支付介绍

## 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](http://kf.qq.com/faq/120911VrYVrA151009JB3i2Q.html" \t "http://kf.qq.com/faq/_blank)| [企业/公司](http://kf.qq.com/faq/120911VrYVrA151013MfYvYV.html" \t "http://kf.qq.com/faq/_blank)| [政府](http://kf.qq.com/faq/161220eaAJjE161220IJn6zU.html" \t "http://kf.qq.com/faq/_blank)| [媒体](http://kf.qq.com/faq/161220IFBJFv161220YnqAbQ.html" \t "http://kf.qq.com/faq/_blank)| [其他类型](http://kf.qq.com/faq/120911VrYVrA151013nYFZ7Z.html" \t "http://kf.qq.com/faq/_blank)。

**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/年。

**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。

## 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>

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

## 3微信支付SDK



添加依赖

|  |
| --- |
| <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*() |

1. MAP转换为XML字符串（自动添加签名）

|  |
| --- |
| WXPayUtil.*generateSignedXml*(param, partnerkey) |

1. XML字符串转换为MAP

|  |
| --- |
| WXPayUtil.*xmlToMap*(result) |

# 微信支付开发

## 1、api接口

场景：用户扫描商户展示在各种场景的二维码进行支付

使用案例：

线下：家乐福超市、7-11便利店、上品折扣线下店等

线上：大众点评网站、携程网站、唯品会、美丽说网站等

开发模式：

模式一：商户在后台给你生成二维码，用户打开扫一扫

模式二：商户后台系统调用微信支付【[统一下单API](https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_1" \t "_blank)】生成预付交易，将接口返回的链接生成二维码，用户扫码后输入密码完成支付交易。注意：**该模式的预付单有效期为2小时，**过期后无法支付。

微信支付：生成xml发送请求

操作模块：service-order

### 引入依赖

|  |
| --- |
| <**dependency**>  <**groupId**>com.github.wxpay</**groupId**>  <**artifactId**>wxpay-sdk</**artifactId**>  <**version**>0.0.3</**version**>  </**dependency**> |

### 1.2 添加配置

在application.properties中添加商户信息

|  |
| --- |
| **spring.redis.host**=**192.168.44.165**  **spring.redis.port**=**6379**  **spring.redis.database**= **0**  **spring.redis.timeout**=**1800000**  **spring.redis.lettuce.pool.max-active**=**20**  **spring.redis.lettuce.pool.max-wait**=**-1**  *#最大阻塞等待时间(负数表示没限制)*  **spring.redis.lettuce.pool.max-idle**=**5**  **spring.redis.lettuce.pool.min-idle**=**0**  *#关联的公众号appid*  **weixin.pay.appid**=**wx74862e0dfcf69954**  *#商户号*  **weixin.pay.partner**=**1558950191**  *#商户key*  **weixin.pay.partnerkey**=**T6m9iK73b0kn9g5v426MKfHQH7X8rKwb** |

### 1.3引入工具类

|  |
| --- |
| @Component  **public class** ConstantPropertiesUtils **implements** InitializingBean {    @Value(**"${weixin.appid}"**)  **private** String **appid**;  @Value(**"${weixin.partner}"**)  **private** String **partner**;  @Value(**"${weixin.partnerkey}"**)  **private** String **partnerkey**;  **public static** String *APPID*;  **public static** String *PARTNER*;  **public static** String *PARTNERKEY*;  @Override  **public void** afterPropertiesSet() **throws** Exception {  *APPID* = **appid**;  *PARTNER* = **partner**;  *PARTNERKEY* = **partnerkey**;  }  } |
| **import** org.apache.http.Consts;  **import** org.apache.http.HttpEntity;  **import** org.apache.http.NameValuePair;  **import** org.apache.http.client.ClientProtocolException;  **import** org.apache.http.client.entity.UrlEncodedFormEntity;  **import** org.apache.http.client.methods.\*;  **import** org.apache.http.conn.ssl.SSLConnectionSocketFactory;  **import** org.apache.http.conn.ssl.SSLContextBuilder;  **import** org.apache.http.conn.ssl.TrustStrategy;  **import** org.apache.http.entity.StringEntity;  **import** org.apache.http.impl.client.CloseableHttpClient;  **import** org.apache.http.impl.client.HttpClients;  **import** org.apache.http.message.BasicNameValuePair;  **import** org.apache.http.ssl.SSLContexts;  **import** org.apache.http.util.EntityUtils;  **import** javax.net.ssl.SSLContext;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.security.KeyStore;  **import** java.security.cert.CertificateException;  **import** java.security.cert.X509Certificate;  **import** java.text.ParseException;  **import** java.util.HashMap;  **import** java.util.LinkedList;  **import** java.util.List;  **import** java.util.Map;  */\*\**  *\* http请求客户端*  *\*/*  **public class** HttpClient {  **private** String **url**;  **private** Map<String, String> **param**;  **private int statusCode**;  **private** String **content**;  **private** String **xmlParam**;  **private boolean isHttps**;  **private boolean isCert** = **false**;  *//证书密码 微信商户号（mch\_id）*  **private** String **certPassword**;  **public boolean** isHttps() {  **return isHttps**;  }  **public void** setHttps(**boolean** isHttps) {  **this**.**isHttps** = isHttps;  }  **public boolean** isCert() {  **return isCert**;  }  **public void** setCert(**boolean** cert) {  **isCert** = cert;  }  **public** String getXmlParam() {  **return xmlParam**;  }  **public void** setXmlParam(String xmlParam) {  **this**.**xmlParam** = xmlParam;  }  **public** HttpClient(String url, Map<String, String> param) {  **this**.**url** = url;  **this**.**param** = param;  }  **public** HttpClient(String url) {  **this**.**url** = url;  }  **public** String getCertPassword() {  **return certPassword**;  }  **public void** setCertPassword(String certPassword) {  **this**.**certPassword** = certPassword;  }  **public void** setParameter(Map<String, String> map) {  **param** = map;  }  **public void** addParameter(String key, String value) {  **if** (**param** == **null**)  **param** = **new** HashMap<String, String>();  **param**.put(key, value);  }  **public void** post() **throws** ClientProtocolException, IOException {  HttpPost http = **new** HttpPost(**url**);  setEntity(http);  execute(http);  }  **public void** put() **throws** ClientProtocolException, IOException {  HttpPut http = **new** HttpPut(**url**);  setEntity(http);  execute(http);  }  **public void** get() **throws** ClientProtocolException, IOException {  **if** (**param** != **null**) {  StringBuilder url = **new** StringBuilder(**this**.**url**);  **boolean** isFirst = **true**;  **for** (String key : **param**.keySet()) {  **if** (isFirst)  url.append(**"?"**);  **else**  url.append(**"&"**);  url.append(key).append(**"="**).append(**param**.get(key));  }  **this**.**url** = url.toString();  }  HttpGet http = **new** HttpGet(**url**);  execute(http);  }  */\*\**  *\* set http post,put param*  *\*/*  **private void** setEntity(HttpEntityEnclosingRequestBase http) {  **if** (**param** != **null**) {  List<NameValuePair> nvps = **new** LinkedList<NameValuePair>();  **for** (String key : **param**.keySet())  nvps.add(**new** BasicNameValuePair(key, **param**.get(key))); *// 参数*  http.setEntity(**new** UrlEncodedFormEntity(nvps, Consts.***UTF\_8***)); *// 设置参数*  }  **if** (**xmlParam** != **null**) {  http.setEntity(**new** StringEntity(**xmlParam**, Consts.***UTF\_8***));  }  }  **private void** execute(HttpUriRequest http) **throws** ClientProtocolException,  IOException {  CloseableHttpClient httpClient = **null**;  **try** {  **if** (**isHttps**) {  **if**(**isCert**) {  FileInputStream inputStream = **new** FileInputStream(**new** File(ConstantPropertiesUtils.*CERT*));  KeyStore keystore = KeyStore.*getInstance*(**"PKCS12"**);  **char**[] partnerId2charArray = **certPassword**.toCharArray();  keystore.load(inputStream, partnerId2charArray);  SSLContext sslContext = SSLContexts.*custom*().loadKeyMaterial(keystore, partnerId2charArray).build();  SSLConnectionSocketFactory sslsf =  **new** SSLConnectionSocketFactory(sslContext,  **new** String[] { **"TLSv1"** },  **null**,  SSLConnectionSocketFactory.***BROWSER\_COMPATIBLE\_HOSTNAME\_VERIFIER***);  httpClient = HttpClients.*custom*().setSSLSocketFactory(sslsf).build();  } **else** {  SSLContext sslContext = **new** SSLContextBuilder()  .loadTrustMaterial(**null**, **new** TrustStrategy() {  *// 信任所有*  **public boolean** isTrusted(X509Certificate[] chain,  String authType)  **throws** CertificateException {  **return true**;  }  }).build();  SSLConnectionSocketFactory sslsf = **new** SSLConnectionSocketFactory(  sslContext);  httpClient = HttpClients.*custom*().setSSLSocketFactory(sslsf)  .build();  }  } **else** {  httpClient = HttpClients.*createDefault*();  }  CloseableHttpResponse response = httpClient.execute(http);  **try** {  **if** (response != **null**) {  **if** (response.getStatusLine() != **null**)  **statusCode** = response.getStatusLine().getStatusCode();  HttpEntity entity = response.getEntity();  *// 响应内容*  **content** = EntityUtils.*toString*(entity, Consts.***UTF\_8***);  }  } **finally** {  response.close();  }  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  httpClient.close();  }  }  **public int** getStatusCode() {  **return statusCode**;  }  **public** String getContent() **throws** ParseException, IOException {  **return content**;  }  } |

### 1.4 添加交易记录接口

#### 1.4.1 添加Mapper

|  |
| --- |
| **public interface** PaymentInfoMapper **extends** BaseMapper<PaymentInfo> {  } |

#### 1.4.2 添加service接口与实现

1、添加PaymentService 类

|  |
| --- |
| **public interface** PaymentService **extends** IService<PaymentInfo> {  */\*\**  *\* 保存交易记录*  *\** ***@param order***  *\** ***@param paymentType*** *支付类型（1：微信 2：支付宝）*  *\*/*  **void** savePaymentInfo(OrderInfo order, Integer paymentType);  } |

2、添加PaymentServiceImpl实现类

|  |
| --- |
| @Service  **public class** PaymentServiceImpl **extends**  ServiceImpl<PaymentInfoMapper, PaymentInfo> **implements** PaymentService {  */\*\**  *\* 保存交易记录*  *\** ***@param orderInfo***  *\** ***@param paymentType*** *支付类型（1：微信 2：支付宝）*  *\*/*  @Override  **public void** savePaymentInfo(OrderInfo orderInfo, Integer paymentType) {  QueryWrapper<PaymentInfo> queryWrapper = **new** QueryWrapper<>();  queryWrapper.eq(**"order\_id"**, orderInfo.getId());  queryWrapper.eq(**"payment\_type"**, paymentType);  Integer count = **baseMapper**.selectCount(queryWrapper);  **if**(count >0) **return**;  *// 保存交易记录*  PaymentInfo paymentInfo = **new** PaymentInfo();  paymentInfo.setCreateTime(**new** Date());  paymentInfo.setOrderId(orderInfo.getId());  paymentInfo.setPaymentType(paymentType);  paymentInfo.setOutTradeNo(orderInfo.getOutTradeNo());  paymentInfo.setPaymentStatus(PaymentStatusEnum.***UNPAID***.getStatus());  String subject = **new** DateTime(orderInfo.getReserveDate()).toString(**"yyyy-MM-dd"**)+**"|"**+orderInfo.getHosname()+**"|"**+orderInfo.getDepname()+**"|"**+orderInfo.getTitle();  paymentInfo.setSubject(subject);  paymentInfo.setTotalAmount(orderInfo.getAmount());  **baseMapper**.insert(paymentInfo);  }  } |

### 1.5 添加支付service接口与实现

1、添加com.atguigu.yygh.order.service.WeixinService类

|  |
| --- |
| **public interface** WeixinService {  */\*\**  *\* 根据订单号下单，生成支付链接*  *\*/*  Map createNative(Long orderId);  } |

2、添加com.atguigu.yygh.order.service.impl.WeixinServiceImpl类

|  |
| --- |
| @Service  **public class** WeixinServiceImpl **implements** WeixinService {  @Autowired  **private** OrderService **orderService**;  @Autowired  **private** PaymentService **paymentService**;  @Autowired  **private** RedisTemplate **redisTemplate**;  */\*\**  *\* 根据订单号下单，生成支付链接*  *\*/*  @Override  **public** Map createNative(Long orderId) {  **try** {  Map payMap = (Map) **redisTemplate**.opsForValue().get(orderId.toString());  **if**(**null** != payMap) **return** payMap;  *//根据id获取订单信息*  OrderInfo order = **orderService**.getById(orderId);  *// 保存交易记录*  **paymentService**.savePaymentInfo(order, PaymentTypeEnum.***WEIXIN***.getStatus());  *//1、设置参数*  Map paramMap = **new** HashMap();  paramMap.put(**"appid"**, ConstantPropertiesUtils.*APPID*);  paramMap.put(**"mch\_id"**, ConstantPropertiesUtils.*PARTNER*);  paramMap.put(**"nonce\_str"**, WXPayUtil.*generateNonceStr*());  String body = order.getReserveDate() + **"就诊"**+ order.getDepname();  paramMap.put(**"body"**, body);  paramMap.put(**"out\_trade\_no"**, order.getOutTradeNo());  *//paramMap.put("total\_fee", order.getAmount().multiply(new BigDecimal("100")).longValue()+"");*  paramMap.put(**"total\_fee"**, **"1"**);  paramMap.put(**"spbill\_create\_ip"**, **"127.0.0.1"**);  paramMap.put(**"notify\_url"**, **"http://guli.shop/api/order/weixinPay/weixinNotify"**);  paramMap.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*(paramMap, ConstantPropertiesUtils.*PARTNERKEY*));  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(**"orderId"**, orderId);  map.put(**"totalFee"**, order.getAmount());  map.put(**"resultCode"**, resultMap.get(**"result\_code"**));  map.put(**"codeUrl"**, resultMap.get(**"code\_url"**));  **if**(**null** != resultMap.get(**"result\_code"**)) {  *//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(orderId.toString(), map, 1000, TimeUnit.***MINUTES***);  }  **return** map;  } **catch** (Exception e) {  e.printStackTrace();  **return new** HashMap<>();  }  }  } |

### 1.6 添加controller方法

|  |
| --- |
| @RestController  @RequestMapping(**"/api/order/weixin"**)  **public class** WeixinController {  @Autowired  **private** WeixinService **weixinPayService**;  */\*\**  *\* 下单 生成二维码*  *\*/*  @GetMapping(**"/createNative/{orderId}"**)  **public** Result createNative(  @ApiParam(name = **"orderId"**, value = **"订单id"**, required = **true**)  @PathVariable(**"orderId"**) Long orderId) {  **return** Result.*ok*(**weixinPayService**.createNative(orderId));  }  } |

## 前端

### 2.1封装api请求

添加/api/order/weixin.js文件

|  |
| --- |
| createNative(orderId) {      return request({      url: `/api/order/weixin/createNative/${orderId}`,      method: 'get'      })  }, |

### 2.2 引入vue-qriously生成二维码

1、安装vue-qriously

npm install vue-qriously

2、引入

在/plugins/myPlugin.js文件添加引入

|  |
| --- |
| **import VueQriously from 'vue-qriously'**  Vue.use(**VueQriously**) |

### 2.3 页面展示

修改/pages/order/show.vue组件

|  |
| --- |
| <template>    <!-- header -->    <div class="nav-container page-component">      <!--左侧导航 #start -->      <div class="nav left-nav">        <div class="nav-item ">          <span class="v-link clickable dark" onclick="javascript:window.location='/user'">实名认证 </span>        </div>        <div class="nav-item selected">          <span class="v-link selected dark" onclick="javascript:window.location='/order'"> 挂号订单 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark" onclick="javascript:window.location='/patient'"> 就诊人管理 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark"> 修改账号信息 </span>        </div>        <div class="nav-item ">          <span class="v-link clickable dark"> 意见反馈 </span>        </div>      </div>      <!-- 左侧导航 #end -->      <!-- 右侧内容 #start -->      <div class="page-container">        <div class="order-detail">          <div class="title"> 挂号详情</div>          <div class="status-bar">            <div class="left-wrapper">              <div class="status-wrapper BOOKING\_SUCCESS">                <span class="iconfont"></span> {{ orderInfo.param.orderStatusString }}              </div>            </div>            <div class="right-wrapper">              <img src="//img.114yygh.com/static/web/code\_order\_detail.png" class="code-img">              <div class="content-wrapper">                <div> 微信<span class="iconfont"></span>关注“预约挂号”</div>                <div class="watch-wrapper"> 快速挂号，轻松就医</div>              </div>            </div>          </div>          <div class="info-wrapper">            <div class="title-wrapper">              <div class="block"></div>              <div>挂号信息</div>            </div>            <div class="info-form">              <el-form ref="form" :model="form">                <el-form-item label="就诊人信息：">                  <div class="content"><span>{{ orderInfo.patientName }}</span></div>                </el-form-item>                <el-form-item label="就诊日期：">                  <div class="content"><span>{{ orderInfo.reserveDate }} {{ orderInfo.reserveTime == 0 ? '上午' : '下午' }}</span></div>                </el-form-item>                <el-form-item label="就诊医院：">                  <div class="content"><span>{{ orderInfo.hosname }} </span></div>                </el-form-item>                <el-form-item label="就诊科室：">                  <div class="content"><span>{{ orderInfo.depname }} </span></div>                </el-form-item>                <el-form-item label="医生职称：">                  <div class="content"><span>{{ orderInfo.title }} </span></div>                </el-form-item>                <el-form-item label="医事服务费：">                  <div class="content">                    <div class="fee">{{ orderInfo.amount }}元                    </div>                  </div>                </el-form-item>                <el-form-item label="挂号单号：">                  <div class="content"><span>{{ orderInfo.outTradeNo }} </span></div>                </el-form-item>                <el-form-item label="挂号时间：">                  <div class="content"><span>{{ orderInfo.createTime }}</span></div>                </el-form-item>              </el-form>            </div>          </div>          <div class="rule-wrapper mt40">            <div class="rule-title"> 注意事项</div>            <div>1、请确认就诊人信息是否准确，若填写错误将无法取号就诊，损失由本人承担；<br>              <span style="color:red">2、【取号】就诊当天需在{{ orderInfo.fetchTime }}在医院取号，未取号视为爽约，该号不退不换；</span><br>              3、【退号】在{{ orderInfo.quitTime }}前可在线退号 ，逾期将不可办理退号退费；<br>              4、预约挂号支持自费患者使用身份证预约，同时支持北京市医保患者使用北京社保卡在平台预约挂号。请于就诊当日，携带预约挂号所使用的有效身份证件到院取号；<br>              5、请注意北京市医保患者在住院期间不能使用社保卡在门诊取号。            </div>          </div>          <div class="bottom-wrapper mt60" v-if="orderInfo.orderStatus == 0 || orderInfo.orderStatus == 1">            <div class="button-wrapper">              <div class="v-button white" @click="cancelOrder()">取消预约</div>            </div>            <div class="button-wrapper ml20" v-if="orderInfo.orderStatus == 0">              <div class="v-button" @click="pay()">支付</div>            </div>          </div>        </div>      </div><!-- 右侧内容 #end -->      <!-- 微信支付弹出框 -->      <el-dialog :visible.sync="dialogPayVisible" style="text-align: left" :append-to-body="true" width="500px" @close="closeDialog">        <div class="container">          <div class="operate-view" style="height: 350px;">            <div class="wrapper wechat">              <div>  **<qriously :value="payObj.codeUrl" :size="220"/>**                <div style="text-align: center;line-height: 25px;margin-bottom: 40px;">                  请使用微信扫一扫<br/>                  扫描二维码支付                </div>              </div>            </div>          </div>        </div>      </el-dialog>    </div><!-- footer -->  </template>  <script>  import '~/assets/css/hospital\_personal.css'  import '~/assets/css/hospital.css'  import orderInfoApi from '@/api/orderInfo'  **import weixinApi from '@/api/weixin'**  export default {    data() {      return {        orderId: null,        orderInfo: {          param: {}        },  **dialogPayVisible: false,**  **payObj: {},**  **timer: null  // 定时器名称**      }    },    created() {      this.orderId = this.$route.query.orderId      this.init()    },    methods: {      init() {        orderInfoApi.getOrders(this.orderId).then(response => {          console.log(response.data);          this.orderInfo = response.data        })      },  **pay() {**  **this.dialogPayVisible = true**  **weixinApi.createNative(this.orderId).then(response => {**  **this.payObj = response.data**  **if(this.payObj.codeUrl == '') {**  **this.dialogPayVisible = false**  **this.$message.error("支付错误")**  **} else {**  **this.timer = setInterval(() => {**  **this.queryPayStatus(this.orderId)**  **}, 3000);**  **}**  **})**  **},**  **queryPayStatus(orderId) {**  **weixinApi.queryPayStatus(orderId).then(response => {**  **if (response.message == '支付中') {**  **return**  **}**  **clearInterval(this.timer);**  **window.location.reload()**  **})**  **},**  **closeDialog() {**  **if(this.timer) {**  **clearInterval(this.timer);**  **}**  **}**    }  }  </script>  <style>    .info-wrapper {      padding-left: 0;      padding-top: 0;    }    .content-wrapper {      color: #333;      font-size: 14px;      padding-bottom: 0;    }    .bottom-wrapper {      width: 100%;    }    .button-wrapper {      margin: 0;    }    .el-form-item {      margin-bottom: 5px;    }    .bottom-wrapper .button-wrapper {      margin-top: 0;    }  </style> |

说明：我们只有轮询查看支付状态，接下来我们处理支付查询接口

### 2.4 添加查询api请求

在/api/weixin.js文件添加方法

|  |
| --- |
| queryPayStatus(orderId) {      return request({          url: `/api/order/weixin/queryPayStatus/${orderId}`,          method: 'get'      })  }, |

## 处理支付结果

### 3.1 支付查询

#### 3.1.1 添加service接口与实现

1、在WeixinService类添加接口

|  |
| --- |
| */\*\**  *\* 根据订单号去微信第三方查询支付状态*  *\*/*  Map queryPayStatus(Long orderId, String paymentType); |

2、在WeixinServiceImpl类添加实现

|  |
| --- |
| @Override  **public** Map queryPayStatus(Long orderId, String paymentType) {  **try** {  OrderInfo orderInfo = **orderService**.getById(orderId);  *//1、封装参数*  Map paramMap = **new** HashMap<>();  paramMap.put(**"appid"**, ConstantPropertiesUtils.*APPID*);  paramMap.put(**"mch\_id"**, ConstantPropertiesUtils.*PARTNER*);  paramMap.put(**"out\_trade\_no"**, orderInfo.getOutTradeNo());  paramMap.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*(paramMap, ConstantPropertiesUtils.*PARTNERKEY*));  client.setHttps(**true**);  client.post();  *//3、返回第三方的数据，转成Map*  String xml = client.getContent();  Map<String, String> resultMap = WXPayUtil.*xmlToMap*(xml);  *//4、返回*  **return** resultMap;  } **catch** (Exception e) {  **return null**;  }  } |

#### 3.1.2 添加controller方法

在WeixinController类添加方法

|  |
| --- |
| @Autowired  **private** PaymentService **paymentService**; |
| @ApiOperation(value = **"查询支付状态"**)  @GetMapping(**"/queryPayStatus/{orderId}"**)  **public** Result queryPayStatus(  @ApiParam(name = **"orderId"**, value = **"订单id"**, required = **true**)  @PathVariable(**"orderId"**) Long orderId) {  *//调用查询接口*  Map<String, String> resultMap = **weixinPayService**.queryPayStatus(orderId, PaymentTypeEnum.***WEIXIN***.name());  **if** (resultMap == **null**) {*//出错*  **return** Result.*fail*().message(**"支付出错"**);  }  **if** (**"SUCCESS"**.equals(resultMap.get(**"trade\_state"**))) {*//如果成功*  *//更改订单状态，处理支付结果*  String out\_trade\_no = resultMap.get(**"out\_trade\_no"**);  **paymentService**.paySuccess(out\_trade\_no, PaymentTypeEnum.***WEIXIN***.getStatus(), resultMap);  **return** Result.*ok*().message(**"支付成功"**);  }  **return** Result.*ok*().message(**"支付中"**);  } |

说明：如果支付成功，我们要更新支付状态、通知医院该预约单已经支付成功

### 3.2 处理支付成功逻辑

#### 3.2.1 添加service接口与实现

1、在PaymentService类添加接口

|  |
| --- |
| */\*\**  *\* 支付成功*  *\*/*  **void** paySuccess(String outTradeNo, Integer paymentType, Map<String, String> paramMap); |

2、在PaymentServiceImpl类添加实现

|  |
| --- |
| */\*\**  *\* 支付成功*  *\*/*  @Override  **public void** paySuccess(String outTradeNo,Integer paymentType, Map<String,String> paramMap) {  PaymentInfo paymentInfo = **this**.getPaymentInfo(outTradeNo, paymentType);  **if** (**null** == paymentInfo) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  **if** (paymentInfo.getPaymentStatus() != PaymentStatusEnum.***UNPAID***.getStatus()) {  **return**;  }  *//修改支付状态*  PaymentInfo paymentInfoUpd = **new** PaymentInfo();  paymentInfoUpd.setPaymentStatus(PaymentStatusEnum.***PAID***.getStatus());  paymentInfoUpd.setTradeNo(paramMap.get(**"transaction\_id"**));  paymentInfoUpd.setCallbackTime(**new** Date());  paymentInfoUpd.setCallbackContent(paramMap.toString());  **this**.updatePaymentInfo(outTradeNo, paymentInfoUpd);  *//修改订单状态*  OrderInfo orderInfo = **orderService**.getById(paymentInfo.getOrderId());  orderInfo.setOrderStatus(OrderStatusEnum.***PAID***.getStatus());  **orderService**.updateById(orderInfo);  *// 调用医院接口，通知更新支付状态*  }  */\*\**  *\* 获取支付记录*  *\*/*  **private** PaymentInfo getPaymentInfo(String outTradeNo, Integer paymentType) {  QueryWrapper<PaymentInfo> queryWrapper = **new** QueryWrapper<>();  queryWrapper.eq(**"out\_trade\_no"**, outTradeNo);  queryWrapper.eq(**"payment\_type"**, paymentType);  **return baseMapper**.selectOne(queryWrapper);  }  */\*\**  *\* 更改支付记录*  *\*/*  **private void** updatePaymentInfo(String outTradeNo, PaymentInfo paymentInfoUpd) {  QueryWrapper<PaymentInfo> queryWrapper = **new** QueryWrapper<>();  queryWrapper.eq(**"out\_trade\_no"**, outTradeNo);  **baseMapper**.update(paymentInfoUpd, queryWrapper);  } |

#### 3.2.2 更新医院支付状态

参考《尚医通API接口文档.docx》业务接口5.2.更新支付状态

|  |
| --- |
| */\*\**  *\* 支付成功*  *\*/*  @Override  **public void** paySuccess(String outTradeNo,Integer paymentType, Map<String,String> paramMap) {  PaymentInfo paymentInfo = **this**.getPaymentInfo(outTradeNo, paymentType);  **if** (**null** == paymentInfo) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  **if** (paymentInfo.getPaymentStatus() != PaymentStatusEnum.***UNPAID***.getStatus()) {  **return**;  }  *//修改支付状态*  PaymentInfo paymentInfoUpd = **new** PaymentInfo();  paymentInfoUpd.setPaymentStatus(PaymentStatusEnum.***PAID***.getStatus());  paymentInfoUpd.setTradeNo(paramMap.get(**"transaction\_id"**));  paymentInfoUpd.setCallbackTime(**new** Date());  paymentInfoUpd.setCallbackContent(paramMap.toString());  **this**.updatePaymentInfo(outTradeNo, paymentInfoUpd);  *//修改订单状态*  OrderInfo orderInfo = **orderService**.getById(paymentInfo.getOrderId());  orderInfo.setOrderStatus(OrderStatusEnum.***PAID***.getStatus());  **orderService**.updateById(orderInfo);  *// 调用医院接口，通知更新支付状态*  SignInfoVo signInfoVo  = **hospitalFeignClient**.getSignInfoVo(orderInfo.getHoscode());  **if**(**null** == signInfoVo) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  Map<String, Object> reqMap = **new** HashMap<>();  reqMap.put(**"hoscode"**,orderInfo.getHoscode());  reqMap.put(**"hosRecordId"**,orderInfo.getHosRecordId());  reqMap.put(**"timestamp"**, HttpRequestHelper.*getTimestamp*());  String sign = HttpRequestHelper.*getSign*(reqMap, signInfoVo.getSignKey());  reqMap.put(**"sign"**, sign);  JSONObject result = HttpRequestHelper.*sendRequest*(reqMap, signInfoVo.getApiUrl()+**"/order/updatePayStatus"**);  **if**(result.getInteger(**"code"**) != 200) {  **throw new** YyghException(result.getString(**"message"**), ResultCodeEnum.***FAIL***.getCode());  }  } |

# 取消预约

## 需求描述

取消订单分两种情况：

1. 未支付取消订单，直接通知医院更新取消预约状态
2. 已支付取消订单，先退款给用户，然后通知医院更新取消预约状态

## 开发微信退款接口

参考文档：https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=9\_4

该接口需要使用证书，详情参考文档并下载证书

### 2.1 配置证书

请下载的证书放在service-order模块/resources/cert文件夹下

在application.properties文件配置证书路径

|  |
| --- |
| **weixin.cert**=  **C:\\yygh\_parent\\service\\service\_order\\src\\main\\resources\\cert\\apiclient\_cert.p12** |

### 2.2 添加获取支付记录接口

退款我们是根据支付记录发起退款的

1、在PaymentService类添加接口

|  |
| --- |
| */\*\**  *\* 获取支付记录*  *\** ***@param orderId***  *\** ***@param paymentType***  *\** ***@return***  *\*/*  PaymentInfo getPaymentInfo(Long orderId, Integer paymentType); |

2、在PaymentServiceImpl类添加实现

|  |
| --- |
| @Override  **public** PaymentInfo getPaymentInfo(Long orderId, Integer paymentType) {  QueryWrapper<PaymentInfo> queryWrapper = **new** QueryWrapper<>();  queryWrapper.eq(**"order\_id"**, orderId);  queryWrapper.eq(**"payment\_type"**, paymentType);  **return baseMapper**.selectOne(queryWrapper);  } |

### 2.3 添加退款记录

#### 2.3 .1 添加mapper

|  |
| --- |
| **public interface** RefundInfoMapper **extends** BaseMapper<RefundInfo> {  } |

#### 2.3 .2 添加service接口与实现

1、添加service接口

|  |
| --- |
| **public interface** RefundInfoService **extends** IService<RefundInfo> {  */\*\**  *\* 保存退款记录*  *\** ***@param paymentInfo***  *\*/*  RefundInfo saveRefundInfo(PaymentInfo paymentInfo);  } |

2、添加service接口实现

|  |
| --- |
| @Service  **public class** RefundInfoServiceImpl **extends** ServiceImpl<RefundInfoMapper, RefundInfo> **implements** RefundInfoService {  @Autowired  **private** RefundInfoMapper **refundInfoMapper**;  @Override  **public** RefundInfo saveRefundInfo(PaymentInfo paymentInfo) {  QueryWrapper<RefundInfo> queryWrapper = **new** QueryWrapper<>();  queryWrapper.eq(**"order\_id"**, paymentInfo.getOrderId());  queryWrapper.eq(**"payment\_type"**, paymentInfo.getPaymentType());  RefundInfo refundInfo = **refundInfoMapper**.selectOne(queryWrapper);  **if**(**null** != refundInfo) **return** refundInfo;  *// 保存交易记录*  refundInfo = **new** RefundInfo();  refundInfo.setCreateTime(**new** Date());  refundInfo.setOrderId(paymentInfo.getOrderId());  refundInfo.setPaymentType(paymentInfo.getPaymentType());  refundInfo.setOutTradeNo(paymentInfo.getOutTradeNo());  refundInfo.setRefundStatus(RefundStatusEnum.***UNREFUND***.getStatus());  refundInfo.setSubject(paymentInfo.getSubject());  *//paymentInfo.setSubject("test");*  refundInfo.setTotalAmount(paymentInfo.getTotalAmount());  **refundInfoMapper**.insert(refundInfo);  **return** refundInfo;  }  } |

### 2.4 添加微信退款接口

1、在WeixinService添加接口

|  |
| --- |
| */\*\*\**  *\* 退款*  *\** ***@param orderId***  *\** ***@return***  *\*/*  Boolean refund(Long orderId); |

2、在WeixinServiceImpl添加实现

|  |
| --- |
| @Override  **public** Boolean refund(Long orderId) {  **try** {  PaymentInfo paymentInfoQuery = **paymentService**.getPaymentInfo(orderId, PaymentTypeEnum.***WEIXIN***.getStatus());  RefundInfo refundInfo = **refundInfoService**.saveRefundInfo(paymentInfoQuery);  **if**(refundInfo.getRefundStatus().intValue() == RefundStatusEnum.***REFUND***.getStatus().intValue()) {  **return true**;  }  Map<String,String> paramMap = **new** HashMap<>(8);  paramMap.put(**"appid"**,ConstantPropertiesUtils.*APPID*); *//公众账号ID*  paramMap.put(**"mch\_id"**,ConstantPropertiesUtils.*PARTNER*); *//商户编号*  paramMap.put(**"nonce\_str"**,WXPayUtil.*generateNonceStr*());  paramMap.put(**"transaction\_id"**,paymentInfoQuery.getTradeNo()); *//微信订单号*  paramMap.put(**"out\_trade\_no"**,paymentInfoQuery.getOutTradeNo()); *//商户订单编号*  paramMap.put(**"out\_refund\_no"**,**"tk"**+paymentInfoQuery.getOutTradeNo()); *//商户退款单号*  *// paramMap.put("total\_fee",paymentInfoQuery.getTotalAmount().multiply(new BigDecimal("100")).longValue()+"");*  *// paramMap.put("refund\_fee",paymentInfoQuery.getTotalAmount().multiply(new BigDecimal("100")).longValue()+"");*  paramMap.put(**"total\_fee"**,**"1"**);  paramMap.put(**"refund\_fee"**,**"1"**);  String paramXml = WXPayUtil.*generateSignedXml*(paramMap,ConstantPropertiesUtils.*PARTNERKEY*);  HttpClient client = **new** HttpClient(**"https://api.mch.weixin.qq.com/secapi/pay/refund"**);  client.setXmlParam(paramXml);  client.setHttps(**true**);  client.setCert(**true**);  client.setCertPassword(ConstantPropertiesUtils.*PARTNER*);  client.post();  *//3、返回第三方的数据*  String xml = client.getContent();  Map<String, String> resultMap = WXPayUtil.*xmlToMap*(xml);  **if** (**null** != resultMap && WXPayConstants.***SUCCESS***.equalsIgnoreCase(resultMap.get(**"result\_code"**))) {  refundInfo.setCallbackTime(**new** Date());  refundInfo.setTradeNo(resultMap.get(**"refund\_id"**));  refundInfo.setRefundStatus(RefundStatusEnum.***REFUND***.getStatus());  refundInfo.setCallbackContent(JSONObject.*toJSONString*(resultMap));  **refundInfoService**.updateById(refundInfo);  **return true**;  }  **return false**;  } **catch** (Exception e) {  e.printStackTrace();  }  **return false**;  } |

### 2.5 完成取消预约

参考《尚医通API接口文档.docx》业务接口5.3.取消预约

#### 2.5.1 添加service接口与实现

1、在OrderService添加接口

|  |
| --- |
| */\*\**  *\* 取消订单*  *\** ***@param orderId***  *\*/*  Boolean cancelOrder(Long orderId); |

2、在OrderServiceImpl添加实现

|  |
| --- |
| @Override  **public** Boolean cancelOrder(Long orderId) {  OrderInfo orderInfo = **this**.getById(orderId);  *//当前时间大约退号时间，不能取消预约*  DateTime quitTime = **new** DateTime(orderInfo.getQuitTime());  **if**(quitTime.isBeforeNow()) {  **throw new** YyghException(ResultCodeEnum.***CANCEL\_ORDER\_NO***);  }  SignInfoVo signInfoVo = **hospitalFeignClient**.getSignInfoVo(orderInfo.getHoscode());  **if**(**null** == signInfoVo) {  **throw new** YyghException(ResultCodeEnum.***PARAM\_ERROR***);  }  Map<String, Object> reqMap = **new** HashMap<>();  reqMap.put(**"hoscode"**,orderInfo.getHoscode());  reqMap.put(**"hosRecordId"**,orderInfo.getHosRecordId());  reqMap.put(**"timestamp"**, HttpRequestHelper.*getTimestamp*());  String sign = HttpRequestHelper.*getSign*(reqMap, signInfoVo.getSignKey());  reqMap.put(**"sign"**, sign);  JSONObject result = HttpRequestHelper.*sendRequest*(reqMap, signInfoVo.getApiUrl()+**"/order/updateCancelStatus"**);  **if**(result.getInteger(**"code"**) != 200) {  **throw new** YyghException(result.getString(**"message"**), ResultCodeEnum.***FAIL***.getCode());  } **else** {  *//是否支付 退款*  **if**(orderInfo.getOrderStatus().intValue() == OrderStatusEnum.***PAID***.getStatus().intValue()) {  *//已支付 退款*  **boolean** isRefund = **weixinService**.refund(orderId);  **if**(!isRefund) {  **throw new** YyghException(ResultCodeEnum.***CANCEL\_ORDER\_FAIL***);  }  }  *//更改订单状态*  orderInfo.setOrderStatus(OrderStatusEnum.***CANCLE***.getStatus());  **this**.updateById(orderInfo);  *//发送mq信息更新预约数 我们与下单成功更新预约数使用相同的mq信息，不设置可预约数与剩余预约数，接收端可预约数减1即可*  OrderMqVo orderMqVo = **new** OrderMqVo();  orderMqVo.setScheduleId(orderInfo.getScheduleId());  *//短信提示*  MsmVo msmVo = **new** MsmVo();  msmVo.setPhone(orderInfo.getPatientPhone());  msmVo.setTemplateCode(**"SMS\_194640722"**);  String reserveDate = **new** DateTime(orderInfo.getReserveDate()).toString(**"yyyy-MM-dd"**) + (orderInfo.getReserveTime()==0 ? **"上午"**: **"下午"**);  Map<String,Object> param = **new** HashMap<String,Object>(){{  put(**"title"**, orderInfo.getHosname()+**"|"**+orderInfo.getDepname()+**"|"**+orderInfo.getTitle());  put(**"reserveDate"**, reserveDate);  put(**"name"**, orderInfo.getPatientName());  }};  msmVo.setParam(param);  orderMqVo.setMsmVo(msmVo);  **rabbitService**.sendMessage(MqConst.***EXCHANGE\_DIRECT\_ORDER***, MqConst.***ROUTING\_ORDER***, orderMqVo);  }  **return true**;  } |

#### 2.5.2 添加controller方法

在OrderApiController添加方法

|  |
| --- |
| @ApiOperation(value = **"取消预约"**)  @GetMapping(**"auth/cancelOrder/{orderId}"**)  **public** Result cancelOrder(  @ApiParam(name = **"orderId"**, value = **"订单id"**, required = **true**)  @PathVariable(**"orderId"**) Long orderId) {  **return** Result.*ok*(**orderService**.cancelOrder(orderId));  } |

### 2.6 修改监听

操作：service-hosp模块

修改HospitalReceiver 类

|  |
| --- |
| @RabbitListener(bindings = @QueueBinding(  value = @Queue(value = MqConst.***QUEUE\_ORDER***, durable = **"true"**),  exchange = @Exchange(value = MqConst.***EXCHANGE\_DIRECT\_ORDER***),  key = {MqConst.***ROUTING\_ORDER***}  ))  **public void** receiver(OrderMqVo orderMqVo, Message message, Channel channel) **throws** IOException {  **if**(**null** != orderMqVo.getAvailableNumber()) {  *//下单成功更新预约数*  Schedule schedule = **scheduleService**.getScheduleId(orderMqVo.getScheduleId());  schedule.setReservedNumber(orderMqVo.getReservedNumber());  schedule.setAvailableNumber(orderMqVo.getAvailableNumber());  **scheduleService**.update(schedule);  } **else** {  ***//取消预约更新预约数***  **Schedule schedule = scheduleService.getScheduleId(orderMqVo.getScheduleId());**  **int availableNumber = schedule.getAvailableNumber().intValue() + 1;**  **schedule.setAvailableNumber(availableNumber);**  **scheduleService.update(schedule);**  }  *//发送短信*  MsmVo msmVo = orderMqVo.getMsmVo();  **if**(**null** != msmVo) {  **rabbitService**.sendMessage(MqConst.***EXCHANGE\_DIRECT\_MSM***, MqConst.***ROUTING\_MSM\_ITEM***, msmVo);  }  } |

## 前端

### 3.1封装api请求

添加/api/weixin.js文件

|  |
| --- |
| cancelOrder(orderId) {      return request({          url: `/api/order/orderInfo/auth/cancelOrder/${orderId}`,          method: 'get'      })  }, |

### 3.2 页面展示

修改/pages/order/show.vue组件

|  |
| --- |
| cancelOrder() {    this.$confirm('确定取消预约吗?', '提示', {      confirmButtonText: '确定',      cancelButtonText: '取消',      type: 'warning'    }).then(() => { // promise      // 点击确定，远程调用      return weixinApi.cancelOrder(this.orderId)    }).then((response) => {      this.$message.success('取消成功')      this.init()    }).catch(() => {      this.$message.info('已取消取消预约')    })  } |