### 支付

### 调用下单接口

思路：

1. Api：<https://pay.weixin.qq.com/wiki/doc/api/app/app.php?chapter=9_1>
2. 需要的是xml格式的数据，组装xml格式数据
3. 签名sign需要使用到前面的参数来生成
4. 最后使用这个xml作为参数去请求微信服务商即可返回结果

*/\*\*  
 \* 调用微信下单，下单后返回的参数至前端，前端使用此参数即可调用支付  
 \* 支付成功后可执行回调函数  
 \** ***@param*** *openId 小程序openId  
 \** ***@param*** *totalFee 总价格  
 \** ***@param*** *orderNo 订单号  
 \** ***@param*** *body 订单名称或其他信息  
 \** ***@return*** *\** ***@throws*** *Exception  
 \*/*public Map<Object, Object> payForOrder(String openId, Double totalFee, String orderNo, String body) throws Exception {  
 //时间戳  
 String times = System.*currentTimeMillis*() + "";  
   
 SortedMap<Object, Object> packageParams = new TreeMap<Object, Object>();  
 //微信小程序ID  
 packageParams.put("appid", appId);  
 //商户ID  
 packageParams.put("mch\_id", mchId);  
 //JSAPI必须要openid  
 packageParams.put("openid", openId);  
 //随机字符串（32位以内）  
 //*todo: 随机生成nonce\_str* packageParams.put("nonce\_str", "ejnshgum39igp6qwpvzwb5jbwk8yigqz");  
 //支付主体名称 自定义  
 packageParams.put("body", body);  
 //订单编号 - 没问题  
 packageParams.put("out\_trade\_no", orderNo);  
 //价格 自定义 单位为分  
 packageParams.put("total\_fee", totalFee.doubleValue() \* 100 + "");  
 //支付返回地址要外网访问的到， localhost不行，调用下面buy方法。（订单存入数据库）  
 packageParams.put("notify\_url", notifyUrl);  
 //这个api有，固定的  
 packageParams.put("trade\_type", "JSAPI");  
 //获取sign - 这个是自己在微信商户设置的32位密钥  
 String sign = PayCommonUtil.*createSign*("UTF-8", packageParams, key);  
 packageParams.put("sign", sign);  
 System.*out*.println(sign);  
 //将参数转成XML  
 String requestXML = PayCommonUtil.*getRequestXml*(packageParams);  
 System.*out*.println(requestXML);  
 //得到含有prepay\_id的XML  
 String resXml = HttpUtil.*postData*(unifiedOrderUrl, requestXML);  
 System.*out*.println(resXml);  
 //解析XML存入Map  
 Map map = XMLUtil.*doXMLParse*(resXml);  
 System.*out*.println(map);  
 //得到prepay\_id  
 String prepay\_id = (String) map.get("prepay\_id");  
 SortedMap<Object, Object> packageP = new TreeMap<>();  
 //！！！注意，这里是appId,上面是appid  
 packageP.put("appId", appId);  
 //时间戳  
 packageP.put("nonceStr", times);  
 //必须把package写成 "prepay\_id="+prepay\_id这种形式  
 packageP.put("package", "prepay\_id=" + prepay\_id);  
 //paySign加密  
 packageP.put("signType", "MD5");  
 packageP.put("timeStamp", (System.*currentTimeMillis*() / 1000) + "");  
 //得到paySign  
 String paySign = PayCommonUtil.*createSign*("UTF-8", packageP, key);  
 packageP.put("paySign", paySign);  
 return packageP;  
}

### 1.2、工具类

#### 1.2.1、 PayCommonUtil（目前只有前两个有用）

import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Date;  
import java.util.Iterator;  
import java.util.List;  
import java.util.Map;  
import java.util.Set;  
import java.util.SortedMap;  
  
public class PayCommonUtil {  
  
 */\*\*  
 \** ***@author*** *\** ***@Description：将请求参数转换为xml格式的string*** *\** ***@param*** *parameters  
 \* 请求参数   
 \** ***@return*** *\*/* public static String getRequestXml(SortedMap<Object, Object> parameters) {  
 StringBuffer sb = new StringBuffer();  
 sb.append("<xml>");  
 Set es = parameters.entrySet();  
 Iterator it = es.iterator();  
 while (it.hasNext()) {  
 Map.Entry entry = (Map.Entry) it.next();  
 String k = entry.getKey().toString();  
 String v = entry.getValue().toString();  
 if ("attach".equalsIgnoreCase(k) || "body".equalsIgnoreCase(k) || "sign".equalsIgnoreCase(k)) {  
 sb.append("<" + k + ">" + v + "</" + k + ">");  
 } else {  
 sb.append("<" + k + ">" + v + "</" + k + ">");  
 }  
 }  
 sb.append("</xml>");  
 return sb.toString();  
 }  
   
 */\*\*  
 \** ***@author*** *\** ***@Description：sign签名*** *\** ***@param*** *characterEncoding  
 \* 编码格式   
 \* 请求参数   
 \** ***@return*** *\*/* public static String createSign(String characterEncoding, SortedMap<Object, Object> packageParams, String API\_KEY) {  
 StringBuffer sb = new StringBuffer();  
 Set es = packageParams.entrySet();  
 Iterator it = es.iterator();  
 while (it.hasNext()) {  
 Map.Entry entry = (Map.Entry) it.next();  
 String k = entry.getKey().toString();  
 String v = entry.getValue().toString();  
 if (null != v && !"".equals(v) && !"sign".equals(k) && !"key".equals(k)) {  
 sb.append(k + "=" + v + "&");  
 }  
 }  
 sb.append("key=" + API\_KEY);  
 String sign = MD5.*MD5Encode*(sb.toString(), characterEncoding).toUpperCase();  
 return sign;  
 }  
 public static String createLinkString(Map<String, String> params) {  
 List<String> keys = new ArrayList<String>(params.keySet());  
 Collections.*sort*(keys);  
 String prestr = "";  
 for (int i = 0; i < keys.size(); i++) {  
 String key = keys.get(i);  
 String value = params.get(key);  
 if (i == keys.size() - 1) {// 拼接时，不包括最后一个&字符   
 prestr = prestr + key + "=" + value;  
 } else {  
 prestr = prestr + key + "=" + value + "&";  
 }  
 }  
 return prestr;  
 }  
  
 */\*\*  
 \* 是否签名正确,规则是:按参数名称a-z排序,遇到空值的参数不参加签名。   
 \** ***@return*** *boolean  
 \*/* public static boolean isTenpaySign(String characterEncoding, SortedMap<Object, Object> packageParams, String API\_KEY) {  
 StringBuffer sb = new StringBuffer();  
 Set es = packageParams.entrySet();  
 Iterator it = es.iterator();  
 while(it.hasNext()) {  
 Map.Entry entry = (Map.Entry)it.next();  
 String k = (String)entry.getKey();  
 String v = (String)entry.getValue();  
 if(!"sign".equals(k) && null != v && !"".equals(v)) {  
 sb.append(k + "=" + v + "&");  
 }  
 }  
  
 sb.append("key=" + API\_KEY);  
  
 //算出摘要   
 String mysign = MD5.*MD5Encode*(sb.toString(), characterEncoding).toLowerCase();  
 String tenpaySign = ((String)packageParams.get("sign")).toLowerCase();  
  
 //System.out.println(tenpaySign + " " + mysign);   
 return tenpaySign.equals(mysign);  
 }  
  
 */\*\*  
 \* 取出一个指定长度大小的随机正整数.   
 \*  
 \** ***@param*** *length  
 \* int 设定所取出随机数的长度。length小于11   
 \** ***@return*** *int 返回生成的随机数。   
 \*/* public static int buildRandom(int length) {  
 int num = 1;  
 double random = Math.*random*();  
 if (random < 0.1) {  
 random = random + 0.1;  
 }  
 for (int i = 0; i < length; i++) {  
 num = num \* 10;  
 }  
 return (int) ((random \* num));  
 }  
  
 */\*\*  
 \* 获取当前时间 yyyyMMddHHmmss   
 \*  
 \** ***@return*** *String  
 \*/* public static String getCurrTime() {  
 Date now = new Date();  
 SimpleDateFormat outFormat = new SimpleDateFormat("yyyyMMddHHmmss");  
 String s = outFormat.format(now);  
 return s;  
 }  
  
 public static boolean verify(String text, String sign, String key, String input\_charset) {  
 text = text + key;  
 String mysign =MD5.*MD5Encode*(text, input\_charset).toUpperCase();  
 System.*out*.println(mysign); System.*out*.println(mysign); System.*out*.println(mysign); System.*out*.println(mysign);  
 if (mysign.equals(sign)) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
}

#### 1.2.2、 XMLUtil（解析XML）

import java.io.ByteArrayInputStream;  
import java.io.InputStream;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.List;  
import java.util.Map;  
  
import org.jdom2.Document;  
import org.jdom2.Element;  
import org.jdom2.input.SAXBuilder;  
  
public class XMLUtil {  
 public static Map doXMLParse(String strxml) throws Exception {  
 strxml = strxml.replaceFirst("encoding=\".\*\"", "encoding=\"UTF-8\"");  
  
 if(null == strxml || "".equals(strxml)) {  
 return null;  
 }  
  
 Map m = new HashMap();  
  
 InputStream in = new ByteArrayInputStream(strxml.getBytes("UTF-8"));  
 SAXBuilder builder = new SAXBuilder();  
 Document doc = builder.build(in);  
 Element root = doc.getRootElement();  
 List list = root.getChildren();  
 Iterator it = list.iterator();  
 while(it.hasNext()) {  
 Element e = (Element) it.next();  
 String k = e.getName();  
 String v = "";  
 List children = e.getChildren();  
 if(children.isEmpty()) {  
 v = e.getTextNormalize();  
 } else {  
 v = XMLUtil.*getChildrenText*(children);  
 }  
  
 m.put(k, v);  
 }  
  
 //关闭流  
 in.close();  
  
 return m;  
 }  
  
 */\*\*  
 \* 获取子结点的xml  
 \** ***@param*** *children  
 \** ***@return*** *String  
 \*/* public static String getChildrenText(List children) {  
 StringBuffer sb = new StringBuffer();  
 if(!children.isEmpty()) {  
 Iterator it = children.iterator();  
 while(it.hasNext()) {  
 Element e = (Element) it.next();  
 String name = e.getName();  
 String value = e.getTextNormalize();  
 List list = e.getChildren();  
 sb.append("<" + name + ">");  
 if(!list.isEmpty()) {  
 sb.append(XMLUtil.*getChildrenText*(list));  
 }  
 sb.append(value);  
 sb.append("</" + name + ">");  
 }  
 }  
  
 return sb.toString();  
 }  
}

#### 1.2.3、HttpUtil（发送请求）

import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.OutputStreamWriter;  
import java.net.URL;  
import java.net.URLConnection;  
  
public class HttpUtil {  
 //private static final Log logger = Logs.get();  
 private final static int *CONNECT\_TIMEOUT* = 5000; // in milliseconds  
 private final static String *DEFAULT\_ENCODING* = "UTF-8";  
  
 public static String postData(String urlStr, String data){  
 return *postData*(urlStr, data, null);  
 }  
  
 public static String postData(String urlStr, String data, String contentType){  
 BufferedReader reader = null;  
 try {  
 URL url = new URL(urlStr);  
 URLConnection conn = url.openConnection();  
 conn.setDoOutput(true);  
 conn.setConnectTimeout(*CONNECT\_TIMEOUT*);  
 conn.setReadTimeout(*CONNECT\_TIMEOUT*);  
 if(contentType != null)  
 conn.setRequestProperty("content-type", contentType);  
 OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), *DEFAULT\_ENCODING*);  
 if(data == null)  
 data = "";  
 writer.write(data);  
 writer.flush();  
 writer.close();  
  
 reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), *DEFAULT\_ENCODING*));  
 StringBuilder sb = new StringBuilder();  
 String line = null;  
 while ((line = reader.readLine()) != null) {  
 sb.append(line);  
 sb.append("\r\n");  
 }  
 return sb.toString();  
 } catch (IOException e) {  
 //logger.error("Error connecting to " + urlStr + ": " + e.getMessage());  
 } finally {  
 try {  
 if (reader != null)  
 reader.close();  
 } catch (IOException e) {  
 }  
 }  
 return null;  
 }  
}