微信支付

1、创建二维码

1、安装 qrcodejs2（注意：安装的是qrcodejs2，不要安装qrcode ---> 会报错）

```
npm install  qrcodejs2 --save
```

2、页面中引入

```
<el-dialog :visible.sync="dialogFormVisible" style="width:800px;margin:0px
auto;" >
  <h1 style="font-size:30px;color:#00B38A">微信扫一扫支付</h1>
  <div id="qrcode" style="width:210px;margin:20px auto;"></div>
</el-dialog>

<script>
import QRCode from 'qrcodejs2'; // 引入qrcodejs

export default {
  name: "Index",
  components: {
    Header,
    Footer,
    QRCode  // 声明组件
  },
  data() {
    return {
      dialogFormVisible: false, // 是否显示登录框，true：显示，false：隐藏
    };
  },
  methods: {
    // 购买课程
    buy(courseid) {
      //alert("购买第【" + courseid + "】门课程成功，加油！");
      this.dialogFormVisible = true; //显示提示框

      // 待dom更新之后再用二维码渲染其内容
      this.$nextTick(function(){
          this.createCode(); // 直接调用会报错：TypeError: Cannot read property
'appendChild' of null
      });
    },
    // 生成二维码
    createCode(){
      // QRCode(存放二维码的dom元素id，二维码的属性参数)
      let qrcode = new QRCode('qrcode',{
         width:200,  // 二维码的宽度
         height:200,  // 二维码的高度
         text:"我爱你中国"  // 二维码中包含的信息
```

```
      });
    },
  },
};
</script>
```

2、准备工作

2.1 名词介绍

参数	说明
appid	微信公众帐号或开放平台APP的唯一标识
partner	商户号（配置文件中的partner：账户）
partnerkey	商户密钥（密码）
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，就是为了保证交易安全

- 如果获得这些信息？
  - 需要注册认证公众号,费用**300元/次**

2.2 获取认证的流程

1) 注册公众号（类型：服务号）

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

2) 认证公众号

公众号认证后才申请微信支付：300元/次

3) 提交材料申请微信支付

登录公众平台，左侧菜单【微信支付】，开始填写资料等待审核，审核时间1~5工作日
这里需要提交的资料有营业执照！

4) 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。

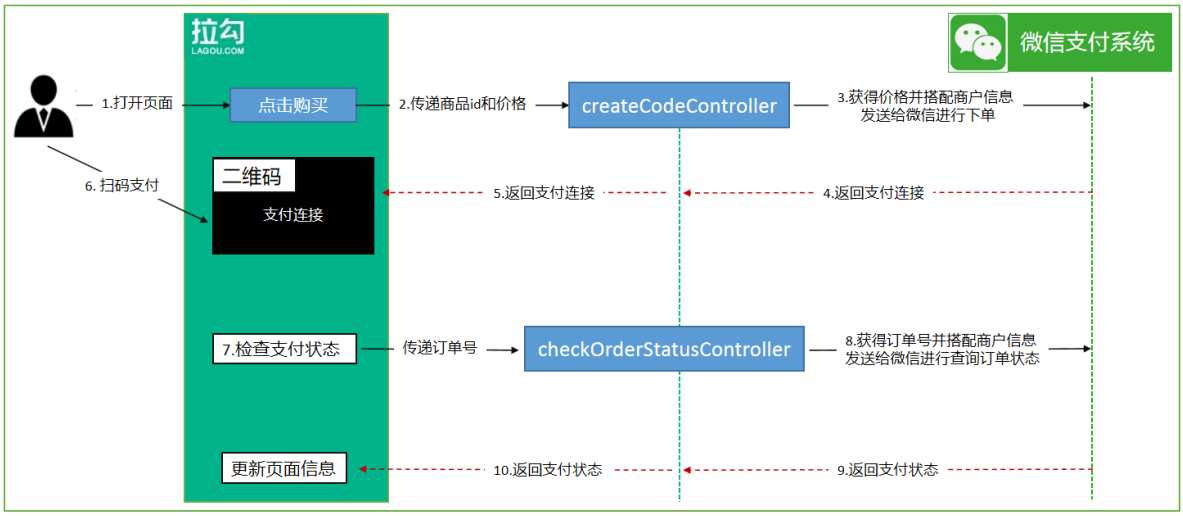
5) 在线签署协议

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。

6) 查看自己的公众号的参数

```
public class PayConfig {
    //企业公众号ID
    public static String appid = "wx8397f8696b538317";
    // 财付通平台的商户帐号
    public static String partner = "1473426802";
    // 财付通平台的商户密钥
    public static String partnerKey = "8A627A4578ACE384017C997F12D68B23";
    // 回调URL
    public static String notifyurl =
"http://a31ef7db.ngrok.io/WeChatPay/WeChatPayNotify";
}
```

3、支付流程



4、工具介绍

4.1 SDK

https://pay.weixin.qq.com/wiki/doc/api/micropay.php?chapter=11_1


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

主要使用sdk中的三个功能：

1、获取随机字符串（生成订单编号）

```
WXPayUtil.generateNonceStr();
```

2、将map转换成xml字符串（自动添加签名）

```
WXPayUtil.generateSignedXml(map,partnerKey);
```

3、将xml字符串转换整map

```
WXPayUtil.xmlToMap(result);
```

4.2 JFinal 框架

- JFinal 是基于Java 语言的极速 web 开发框架，其核心设计目标是开发迅速、代码量少、学习简单、功能强大、轻量级、易扩展
- 取代HttpClient

```
<dependency>
    <groupId>com.jfinal</groupId>
    <artifactId>jfinal</artifactId>
    <version>3.5</version>
</dependency>
```

5、构建二维码

- Course.vue

```
<script>
//  生成二维码
    createCode () {
      this.axios
        .get("http://localhost:80/order/createCode", {
          params: {
            courseid: this.course.id,  // 课程编号
            courseid: this.course.courseName, // 课程名称
            price: this.course.discounts,  // 优惠价，非原价
          },
        })
        .then((result) => {
          console.log(result);
          let qrcode = new QRCode('qrcode',{
            width:200,
            height:200,
            text:result.data.code_url  // 将支付连接嵌入到二维码中
          });
        })
        .catch((error) => {
          this.$message.error("二维码生成失败!");
        });
    }
</script>
```

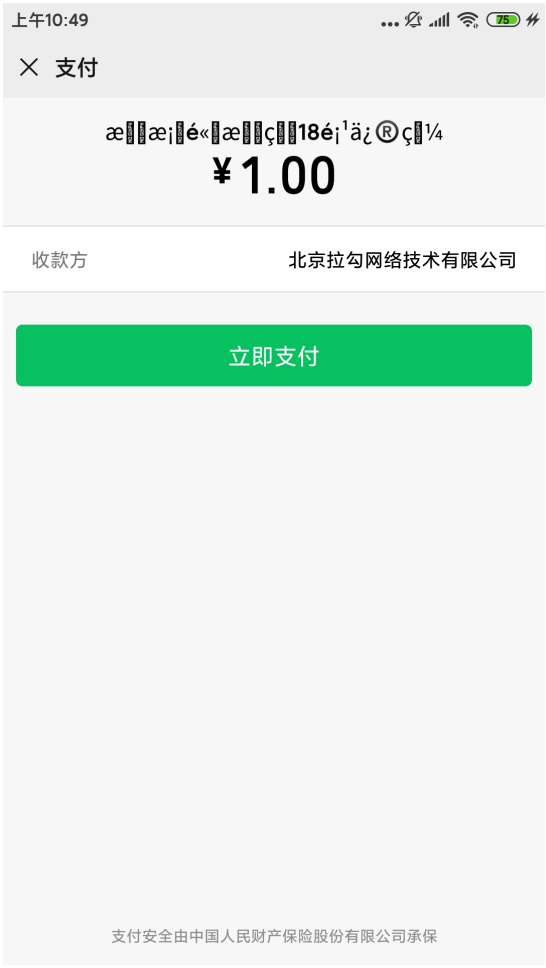
- 支付配置

```
public class PayConfig {
    //企业公众号ID
    public static String appid = "wx8397f8696b538317";
    // 财付通平台的商户帐号
    public static String partner = "1473426802";
    // 财付通平台的商户密钥
    public static String partnerKey = "8A627A4578ACE384017C997F12D68B23";
    // 回调URL
    public static String notifyurl =
"http://a31ef7db.ngrok.io/WeChatPay/WeChatPayNotify";
}
```

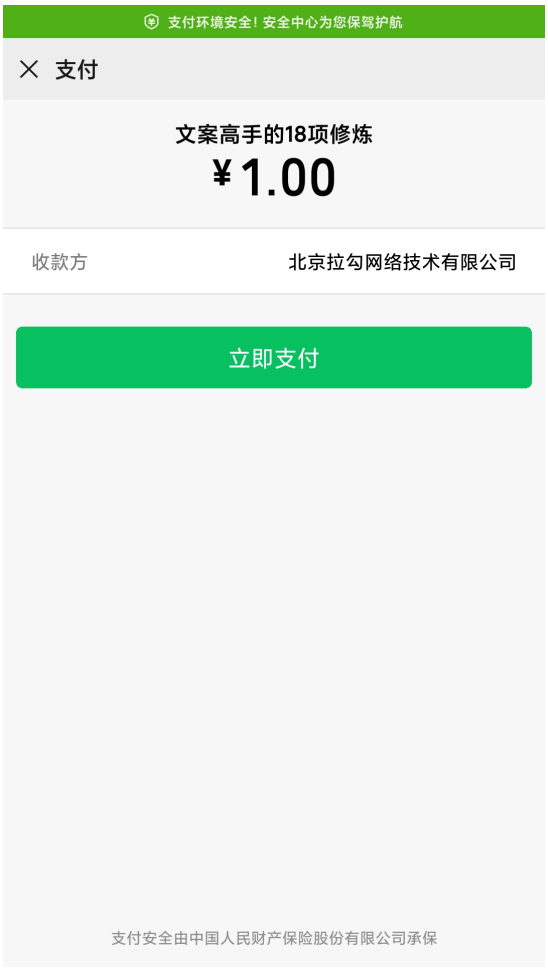
- createCodeController

```
import com.alibaba.fastjson.JSON;
import com.github.wxpay.sdk.WXPayUtil;
import com.jfinal.kit.HttpKit;
import commons.PayConfig;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;
import java.util.Map;
```

```
@RestController
@RequestMapping("order")
public class PayAction {

    @GetMapping("createCode")
    public Object createCode(String courseid,String coursename, String price)
throws Exception {

        //1.编写商户信息
        Map<String,String> mm = new HashMap();
        mm.put("appid",PayConfig.appid); //公众账号ID
        mm.put("mch_id",PayConfig.mchid);//商户号
        mm.put("nonce_str",WXPayUtil.generateNonceStr());//随机字符串
        mm.put("body",coursename); //商品名称
        String orderId = WXPayUtil.generateNonceStr();
        System.out.println("订单编号 = "+orderId);
        mm.put("out_trade_no",orderId); //商户订单号
        mm.put("total_fee",price+""); //订单金额,单位分
        mm.put("spbill_create_ip","127.0.0.1"); //终端IP
        mm.put("notify_url",PayConfig.notifyurl); //通知地址
        mm.put("trade_type","NATIVE"); //交易类型
        System.out.println("发送的map = "+mm.toString());

        //2.生成数字签名,并把上面的map转换成xml格式
        String xml = WXPayUtil.generateSignedXml(mm,PayConfig.partnerKey);
        System.out.println("转换后的xml = "+xml);

        //3.将数据发送给微信后台,并得到微信后台返回的数据
        String url = "https://api.mch.weixin.qq.com/pay/unifiedorder";
        String result = HttpKit.post(url,xml);
        System.out.println("返回的xml = "+result); //如果报错：<![CDATA[签名错误]]>
商户四要素的原因，重置商户API密钥。

        //4.后台返回的xml格式，转成map，并添加两个参数
        Map<String,String> resultMap = WXPayUtil.xmlToMap(result);
        resultMap.put("orderId",orderId);
        resultMap.put("money",price+"");

        //5.将map返回给浏览器
        return resultMap;
    }
}
```




```
coursename = new String(coursename.getBytes("ISO-8859-1"),"UTF-8");
```



6、检查支付状态

```
<!-- 微信支付二维码-->
<el-dialog :visible.sync="dialogFormVisible" style="width:800px;margin:0px
auto;" >
  <h1 style="font-size:30px;color:#00B38A" >微信扫一扫支付</h1>
  <div id="qrcode" style="width:210px;margin:20px auto;"></div>
  <h2 id="statusText"></h2>
  <p id="timeClose"></p>
</el-dialog>
<script>
data() {
    return {
      time:null // 定时器
    };
},
// 生成二维码
createCode(){
  // 去获取支付连接
  this.axios
    .get("http://localhost:80/order/createCode",{
      params:{
        courseid: this.course.id,
        coursename: this.course.courseName,
        price:this.course.discounts,
      }
    })
    .then((result) => {
        console.log("订单号1："+result.data.out_trade_no);
        // QRCode(存放二维码的dom元素id，二维码的属性参数)
        let qrcode = new QRCode('qrcode',{
           width:200,
           height:200,
           text:result.data.code_url  // 将返回的数据嵌入到二维码中
       });

       // 检查订单状态
       this.axios
          .get("http://localhost:80/order/checkOrderStatus",{
            params:{
              orderId: this.orderId
            }
          })
          .then((result) => {
              console.log(result);
              if(result.data.trade_state=="SUCCESS"){
     document.getElementById("statusText").innerHTML = "<i style='color:#00B38A'
class='el-icon-success'></i> 支付成功！";
                  let s = 3;
                  this.closeQRForm(s);
              }
          })
```

```
            .catch( (error)=>{
                this.$message.error("查询订单状态失败！");
            });
    })
    .catch( (error)=>{
      this.$message.error("生成二维码失败！");
    });
},
// 倒计时关闭二维码窗口
closeQRForm(s){
  let that = this;
  that.time= setInterval(function(){
      document.getElementById("timeClose").innerHTML = "("+ s-- +")秒后关闭本窗口！";
      if(s == 0){
        clearInterval(that.time); // 停止计时
        that.dialogFormVisible = false; // 关闭
        that.isBuy = true; // 解锁购买状态
      }
  },1000);
},
</script>
```

```
@GetMapping("createCode")
public Object createCode(String courseid,String coursename,String price) throws Exception {
    // 省略...
    // 查询订单状态需要订单编号，所以将订单编号保存并返回给前端
    resultMap.put("out_trade_no",mm.get("out_trade_no"));
    return resultMap;
}
```

```
@GetMapping("checkOrderStatus")
public Object checkOrderStatus(String orderId) throws Exception {
    //1.编写商户信息
    Map<String,String> mm = new HashMap();
    mm.put("appid",PayConfig.appid); //公众账号ID
    mm.put("mch_id",PayConfig.partner);//商户号
    mm.put("out_trade_no",orderId);//订单编号
    mm.put("nonce_str",WXPayUtil.generateNonceStr()); //随机字符串
    System.out.println(mm);
    //2.生成数字签名,并把上面的map转换成xml格式
    String xml = WXPayUtil.generateSignedXml(mm, PayConfig.partnerKey);
    System.out.println(xml);

    //3.将数据发送给微信后台,并得到微信后台返回的数据
    String url = "https://api.mch.weixin.qq.com/pay/orderquery";

    //第一次询问时间
    long beginTime = System.currentTimeMillis();

    while(true) { //不停的去微信后台询问是否支付
```

```
        String result = HttpKit.post(url, xml);
        System.out.println(result);//报错：<![CDATA[签名错误]]>

        //4.后台返回的xml格式，转成map，并添加两个参数
        Map<String, String> resultMap = WXPayUtil.xmlToMap(result);

        //5.将map转成json并返回给浏览器
        //已经成功支付，停止询问
        if(resultMap.get("trade_state").equalsIgnoreCase("success")){
            return resultMap;
        }
        //超过30秒未支付，停止询问
        if(System.currentTimeMillis() - beginTime > 30000){
            return resultMap;
        }
        Thread.sleep(3000); //每隔3秒，询问一次微信后台
    }
}
```

7、保存订单并更新状态

```
// 检查支付状态
this.axios
 .get("http://localhost:80/order/checkOrderStatus",{
  params:{
    orderId: result.data.orderId // 传递 订单编号 进行查询
  }
})
.then((result) => {

  if(result.data.trade_state=="SUCCESS"){
    document.getElementById("statusText").innerHTML = "<i style='color:#00B38A'
class='el-icon-success'></i> 支付成功！";
    // 支付成功
    this.updateOrder(20);
  }
  /*
  else if(result.data.trade_state=="NOTPAY"){
    document.getElementById("statusText").innerHTML = "<i style='color:#00B38A'
class='el-icon-success'></i> 未支付！";
    this.updateOrder(10);
  }
  */

  // 3秒后关闭二维码窗口
  let s = 3;
  this.closeQRForm(s);
})
.catch( (error)=>{
  this.$message.error("查询订单失败！");
});
```

```
// 更新订单的状态
updateOrder(statusCode){
  return this.axios
    .get("http://localhost:80/order/updateOrder",{
      params:{
        orderNo:this.orderNo,
        status:statusCode,
      }
    })
    .then((result) => {
       console.log("更新订单【"+this.orderNo+"】状态：" + statusCode);
    }).catch( (error)=>{
       this.$message.error("更新订单失败！");
    });
},
```

- web消费方

```
@GetMapping("updateOrder")
public Integer updateOrder(String orderNo , Integer status) {
    System.out.println("订单编号 = " + orderNo);
    System.out.println("状态编码 = " + status);
    Integer integer = orderService.updateOrder(orderNo, status);
    System.out.println("订单更新 = " + integer);
    return integer;
}
```