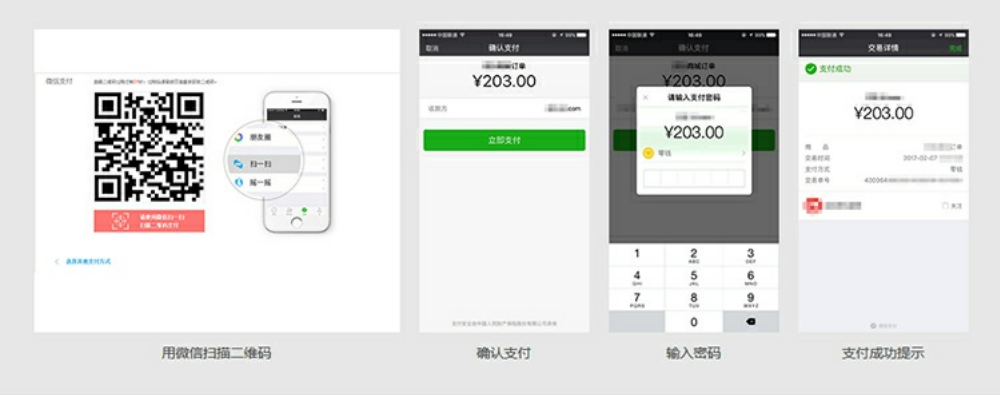
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=126 sentences=280, words=1746

1. 微信支付

- 扫码支付是指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"来完成支付。
- 适用于PC网站支付、实体店单品等场景。



2. 申请流程

2.1 注册公众号

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)
- 企业/公司 (http://kf.qq.com/faq/120911VrYVrA151013MfYvYV.html)
- 政府 (http://kf.qq.com/faq/120911VrYVrA15100973ABZz.html)
- 媒体 (http://kf.qq.com/faq/120911VrYVrA151013aMNfeQ.html)
- 其他类型 (http://kf.qq.com/faq/120911VrYVrA151013nYFZ7Z.html)|

2.2 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 查看认证流程 (http://kf.qq.com/product/weixinmp.html#hid=97)

2.3 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48小时内。

2.4 开户成功，进行账户验证

资料审核通过后，开户信息会通过邮件、公众号发送给联系人，请按照指引填写财付通备付金汇入的随机金额，完成账户验证。（查看验证方法）

2.5 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。点此提前预览协议内容。

2.6 启动设计和开发

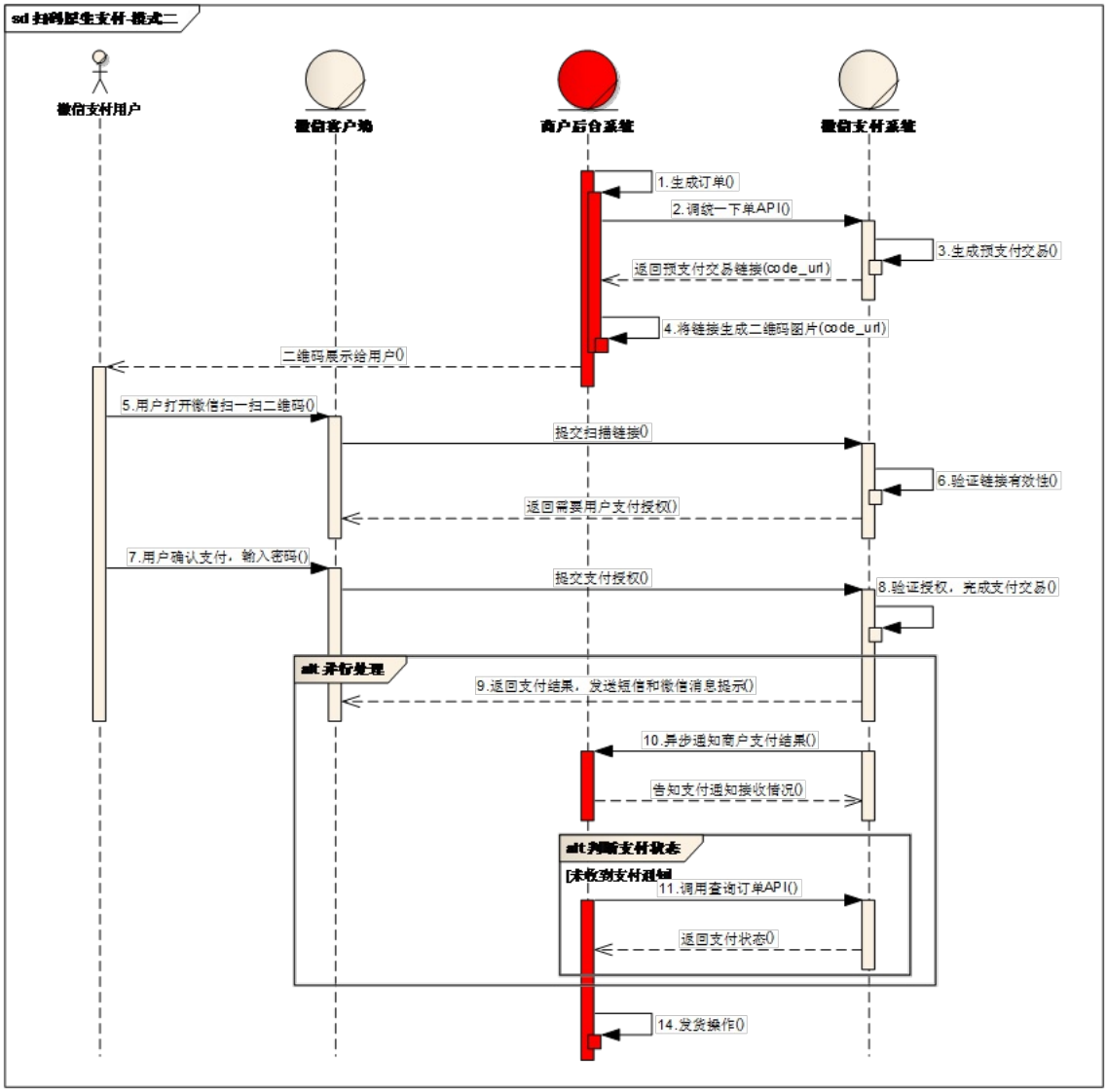
支付接口已获得，可根据开发文档 (https://pay.weixin.qq.com/wiki/doc/api/index.html)进行开发，也可了解成功案例界面示意及素材。

3. 准备素材

- 扫码支付 (https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=6_1)
- APPID 公众平台 (https://mp.weixin.qq.com)(1959583119)->右上角->开发者ID->AppID
- 商户号码和配置回调链接商户平台 (https://pay.weixin.qq.com)->产品中心->开发配置->商户信息->商户号
- 配置APPID 商户平台 (https://pay.weixin.qq.com)->产品中心->APPID授权管理
- API密钥商户平台 (https://pay.weixin.qq.com)->账户中心->API安全->设置API密钥

4.接入流程

- 商户后台系统先调用微信支付的统一下单接口
- 微信后台系统返回链接参数code_url
- 商户后台系统将code_url值生成二维码图片
- 用户使用微信客户端扫码后发起支付
- 注意：code_url有效期为2小时，过期后扫码不能再发起支付。



业务流程说明：

- 1）商户后台系统根据用户选购的商品生成订单。
- 2）用户确认支付后调用微信支付统一下单API (https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_1)生成预支付交易；
- 3）微信支付系统收到请求后生成预支付交易单，并返回交易会话的二维码链接code_url。
- 4）商户后台系统根据返回的code_url生成二维码。
- 5）用户打开微信"扫一扫"扫描二维码，微信客户端将扫码内容发送到微信支付系统。
- 6）微信支付系统收到客户端请求，验证链接有效性后发起用户支付，要求用户授权。
- 7）用户在微信客户端输入密码，确认支付后，微信客户端提交授权。
- 8）微信支付系统根据用户授权完成支付交易。
- 9）微信支付系统完成支付交易后给微信客户端返回交易结果，并将交易结果通过短信、微信消息提示用户。微信客户端展示支付交易结果页面。
- 10）微信支付系统通过发送异步消息通知商户后台系统支付结果。商户后台系统需回复接收情况，通知微信后台系统不再发送该单的支付通知。
- 11）未收到支付通知的情况，商户后台系统调用查询订单API (https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_2)
- 12）商户确认订单已支付后给用户发货。

5.绘制页面

5.1 生成项目

```
$ npm i egg-init -g
$ egg-init zhufeng-wxpay --type=simple
$ cd zhufeng-wxpay
$ npm i
$ npm run dev
$ open localhost:7001

ssh -vnNT -R 7689:localhost:7001 root@47.105.88.180
```

5.2 支持模版

5.2.1 package.json

```
yarn add egg-view-ejs
```

5.2.2 app/controller/home.js

app/controller/home.js

```
   async checkout() {
-
+      const { ctx, app } = this;
+      await ctx.render('checkout');
   }
 }
```

5.2.3 checkout.html

app/view/checkout.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <script src="https://cdn.bootcss.com/jquery/3.2.1/jquery.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous">script>
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous">script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous">script>
    <title>微信支付title>
head>
<body>
        <div class="container">
            <div class="row justify-content-center">
                <div class="col-4">
                    <div class="card text-center mt-5">
                        <div class="card-header">
                        <h3>微信支付h3>
                        div>
                        <img src="/public/images/qrcode.png" class="card-img-top">
                        <div class="card-body">
                            <img src="/public/images/scan.png" alt="" class="w-100">
                        div>
                    div>
                div>
            div>
        div>
body>
html>
```

5.2.4 config.default.js

config/config.default.js

```
  config.view={
    defaultViewEngine: 'ejs',
    defaultExtension:'.html',
    mapping: {
       '.html':'ejs'
    }
  }
```

5.2.5 plugin.js

config/plugin.js

```
exports.ejs = {
    enable: true,
    package: 'egg-view-ejs',
};
```

5.3 支付和通知

5.3.1 package.json

```
   "dependencies": {
+    "axios": "^0.18.0",
+    "moment": "^2.22.2",
+    "qrcode": "^1.2.2",
+    "randomstring": "^1.1.5",
+    "uuid": "^3.3.2",
+    "xml-js": "^1.6.7"
   }
```

5.3.2 app/router.js

app/router.js

```
  router.post('/wxpay/notify',controller.home.notify);
```

5.3.3 home.js

app/controller/home.js

```
const Controller = require('egg').Controller;
const moment = require('moment');
const randomstring = require('randomstring');
const querystring = require('querystring');
const crypto = require('crypto');
const xmljs = require('xml-js');
const axios = require('axios');
const qrcode = require('qrcode');
class HomeController extends Controller {
  async index() {
    this.ctx.body = 'hi, egg';
  }
  wxSign(order, key,logger) {

    let sortedOrder = Object.keys(order).sort().reduce((memo, key) => {
```

```
    memo[key] = order[key];
    return memo;
  }, {});
  logger.info('排过序的订单', sortedOrder);

  const stringifiedOrder = querystring.stringify(sortedOrder, null, null, {
    encodeURIComponent: querystring.unescape
  });
  logger.info('不带key的订单字符串', stringifiedOrder);

  const stringifiedOrderWithKey = `${stringifiedOrder}&key=${key}`;
  logger.info('带key的订单字符串', stringifiedOrderWithKey);

  const sign = crypto.createHash('md5').update(stringifiedOrderWithKey).digest('hex').toUpperCase();
  return sign;
}
async checkout() {
  const {
    ctx,
    app
  } = this;
  const {
    logger
  } = ctx;
  const {
    wxpay = {}
  } = app.config;

  const appid = wxpay.appid;

  const mch_id = wxpay.mch_id;

  const notify_url = wxpay.notify_url;

  const key = wxpay.key;

  const unifiedorder = wxpay.unifiedorder;

  const out_trade_no = moment().local().format('YYYYMMDDhhmmss');

  const body = '珠峰培训';

  const total_fee = 1;

  const trade_type = 'NATIVE';

  const product_id = 100;

  const nonce_str = randomstring.generate(32);

  let order = {
    appid,
    mch_id,
    out_trade_no,
    body,
    total_fee,
    product_id,
    notify_url,
    nonce_str,
    trade_type
  }
  const sign = this.wxSign(order, key,logger);
  logger.info('签名', sign);

  const xmlOrder = xmljs.js2xml({
    xml: {
      ...order,
      sign
    }
  }, {
    compact: true
  });
  logger.info('XML订单', xmlOrder);

  const unifiedorderResponse = await axios.post(unifiedorder, xmlOrder);
  logger.info('统一下单响应', unifiedorderResponse.data);

  const _prepay = xmljs.xml2js(unifiedorderResponse.data, {
    compact: true,
    cdataKey: 'value',
    textKey:'value'
  });
  logger.info('_prepay', _prepay);

  const prepay = Object.entries(_prepay.xml).reduce((memo, [key, value]) => {
    memo[key]=value.value;
    return memo;
  }, {});

  logger.info('prepay', prepay);
  const {
    code_url
  } = prepay;
  const qrcodeUrl = await qrcode.toDataURL(code_url, {
    width: 300
  });
  await ctx.render('checkout', {
    qrcodeUrl
  });
}
parseXML(req) {
    return new Promise(function(resolve,reject){
        let buffers = [];
        req.on('data', function(data) {
```

```
            buffers.push(data);
        });
        req.on('end', function() {
            let ret = Buffer.concat(buffers);
            resolve(ret.toString());
        });
    });
  }
  async notify() {
    const {ctx,app} = this;
      const { logger } = ctx;
      const {
        wxpay = {}
      } = app.config;
      const key = wxpay.key;
    let body = await this.parseXML(ctx.req);
    logger.debug('request',body);
    const _payment = xmljs.xml2js(body, {
      compact: true,
      cdataKey: 'value',
      textKey:'value'
    });
    logger.info('_payment', _payment);
    const payment = Object.entries(_payment.xml).reduce((memo, [key, value]) => {
      memo[key] = value.value;
      return memo;
    }, {});
    logger.info('payment', payment);
    const paymentSign = payment.sign;
    logger.debug('paymentSign', paymentSign);
    delete payment.sign;
    const mySign = this.wxSign(payment, key,logger);
    logger.debug('mySign', mySign);
    const return_code = paymentSign === mySign ? 'SUCCESS' : "FAIL";
    const reply = {
      xml: {
        return_code
      }
    }
    const ret = xmljs.js2xml(reply, {
      compact: true
    });
    logger.debug('通知返回', ret);
    ctx.body = ret;
  }
}

module.exports = HomeController;
```

5.3.4 checkout.html

app/view/checkout.html

```
-
+
```

5.3.5 config.default.js

config/config.default.js

```
  config.logger={
    level: 'DEBUG',
    allowDebugAtProd: true,
    consoleLevel: 'DEBUG'
  }
  exports.security = {
    csrf: {
      enable: false,
      ignoreJSON: true
    }
  };
  config.wxpay={
    appid: '',
    mch_id: '',
    key: '',
    notify_url: 'http://front.zhufengpeixun.cn/wxpay/notify',
    unifiedorder:'https://api.mch.weixin.qq.com/pay/unifiedorder'
  }
```

5.4 查询结果

5.4.1 app/router.js

```
router.get('/wxpay/query',controller.home.query);
```

5.4.2 result.html

app/view/result.html

```
<body>
    <div class="container">
        <div class="jumbotron mt-5 text-center">
            <h1><%=trade_state_desc%>h1>
        div>
    div>
body>
```

5.4.3 config.default.js

config/config.default.js

```
+    unifiedorder: 'https://api.mch.weixin.qq.com/pay/unifiedorder',
+    orderquery:'https://api.mch.weixin.qq.com/pay/orderquery'
```

5.4.4 app/controller/home.js

home.js

```
;

const Controller = require('egg').Controller;
const moment = require('moment');
const randomstring = require('randomstring');
const querystring = require('querystring');
const crypto = require('crypto');
const transformer = require('xml-js');
const axios = require('axios');
const qrcode = require('qrcode');
class HomeController extends Controller {
  async index() {
    this.ctx.body = 'hi, egg';
  }
  wxSign(order, key,logger) {

    let sortedOrder = Object.keys(order).sort().reduce((memo, key) => {
      memo[key] = order[key];
      return memo;
    }, {});
    logger.info('排过序的订单', sortedOrder);

    const stringifiedOrder = querystring.stringify(sortedOrder, null, null, {
      encodeURIComponent: querystring.unescape
    });
    logger.info('不带key的订单字符串', stringifiedOrder);

    const stringifiedOrderWithKey = `${stringifiedOrder}&key=${key}`;
    logger.info('带key的订单字符串', stringifiedOrderWithKey);

    const sign = crypto.createHash('md5').update(stringifiedOrderWithKey).digest('hex').toUpperCase();
    return sign;
  }
  xml2js(data,logger) {
    const _result = transformer.xml2js(data, {
      compact: true,
      cdataKey: 'value',
      textKey:'value'
    });
    logger.info('_result', _result);

    const result = Object.entries(_result.xml).reduce((memo, [key, value]) => {
      memo[key]=value.value;
      return memo;
    },{});
    logger.info('result', result);
    return result;
  }
  async checkout() {
    const {
      ctx,
      app
    } = this;
    const {
      logger
    } = ctx;
    const {
      wxpay = {}
    } = app.config;

    const appid = wxpay.appid;

    const mch_id = wxpay.mch_id;

    const notify_url = wxpay.notify_url;

    const key = wxpay.key;

    const unifiedorder = wxpay.unifiedorder;

    const out_trade_no=moment().local().format('YYYYMMDDhhmmss');
    ctx.session.out_trade_no=out_trade_no;

    const body = '珠峰培训';

    const total_fee = 1;

    const trade_type = 'NATIVE';

    const product_id = 100;

    const nonce_str = randomstring.generate(32);

    let order = {
      appid,
      mch_id,
      out_trade_no,
      body,
      total_fee,
      product_id,
      notify_url,
      nonce_str,
      trade_type
    }
    const sign = this.wxSign(order, key,logger);
    logger.info('签名', sign);

    const xmlOrder = transformer.js2xml({
      xml: {
        ...order,
```

```
      sign
    }
  }, {
    compact: true
  });
  logger.info('XML订单', xmlOrder);

  const unifiedorderResponse = await axios.post(unifiedorder, xmlOrder);
  logger.info('统一下单响应', unifiedorderResponse.data);
  const prepay=this.xml2js(unifiedorderResponse.data,logger);
  logger.info('prepay', prepay);
  const {
    code_url
  } = prepay;
  const qrcodeUrl = await qrcode.toDataURL(code_url, {
    width: 300
  });
  await ctx.render('checkout', {
    qrcodeUrl
  });
}
parseXML(req) {
    return new Promise(function(resolve,reject){
        let buffers = [];
        req.on('data', function(data) {
            buffers.push(data);
        });
        req.on('end', function() {
            let ret = Buffer.concat(buffers);
            resolve(ret.toString());
        });
    });
}
async notify() {
  const {ctx,app} = this;
    const { logger } = ctx;
    const {
      wxpay = {}
    } = app.config;
    const key = wxpay.key;
  let body = await this.parseXML(ctx.req);
  logger.debug('request',body);
  const payment=this.xml2js(body,logger);
  logger.info('payment', payment);
  const paymentSign = payment.sign;
  logger.debug('paymentSign', paymentSign);
  delete payment.sign;
  const mySign = this.wxSign(payment, key,logger);
  logger.debug('mySign', mySign);
  const return_code = paymentSign === mySign ? 'SUCCESS' : "FAIL";
  const reply = {
    xml: {
      return_code
    }
  }
  const ret = transformer.js2xml(reply, {
    compact: true
  });
  logger.debug('通知返回', ret);
  ctx.body = ret;
}

async query() {
  const {
    ctx,
    app
  } = this;
  const {
    logger
  } = ctx;
  const {
    wxpay = {}
  } = app.config;

  const appid = wxpay.appid;

  const mch_id = wxpay.mch_id;

  const key = wxpay.key;

  const orderquery=wxpay.orderquery;

  const nonce_str = randomstring.generate(32);
  let query={
    appid,
    mch_id,
    out_trade_no: ctx.session.out_trade_no,
    nonce_str
  }

  let sign=this.wxSign(query,key,logger);
  logger.info('签名', sign);

  const xmlQuery = transformer.js2xml({
    xml: {
      ...query,
      sign
    }
  }, {
    compact: true
  });
  logger.info('XML查询条件', xmlQuery);

  const orderqueryResponse = await axios.post(orderquery, xmlQuery);
```

```
    logger.info('订单查询响应',orderqueryResponse.data);
    const queryResult =this.xml2js(orderqueryResponse.data,logger);
    logger.info('queryResult',queryResult);
    await ctx.render('result',queryResult);
  }
}
module.exports = HomeController;
```

参考资料

- 扫码支付接入方法指引 (http://kf.qq.com/faq/170116AziqYV1701162eyAzA.html)
- 登录微信支付 (https://pay.weixin.qq.com)
- 微信支付WIKI (https://pay.weixin.qq.com/wiki/doc/api/index.html)
- 微信帮助文档 (http://kf.qq.com/product/weixinmp.html)
- 登录微信公众账号 (https://mp.weixin.qq.com)
- 微信支付接入教程 (https://jz.fkw.com/blog/660)
- 统一下单API (https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_1)
- 查询订单API (https://pay.weixin.qq.com/wiki/doc/api/native.php?chapter=9_2)
- xml-js (https://www.npmjs.com/package/xml-js)