微服务项目之支付服务

一、支付简介

1.1 支付演变

汇票

网盾

在线支付

闪付

支付：安全

产品：稳定

1.2 支付宝支付

https://open.alipay.com/developmentAccess/developmentAccess.htm

json数据

使用步骤：

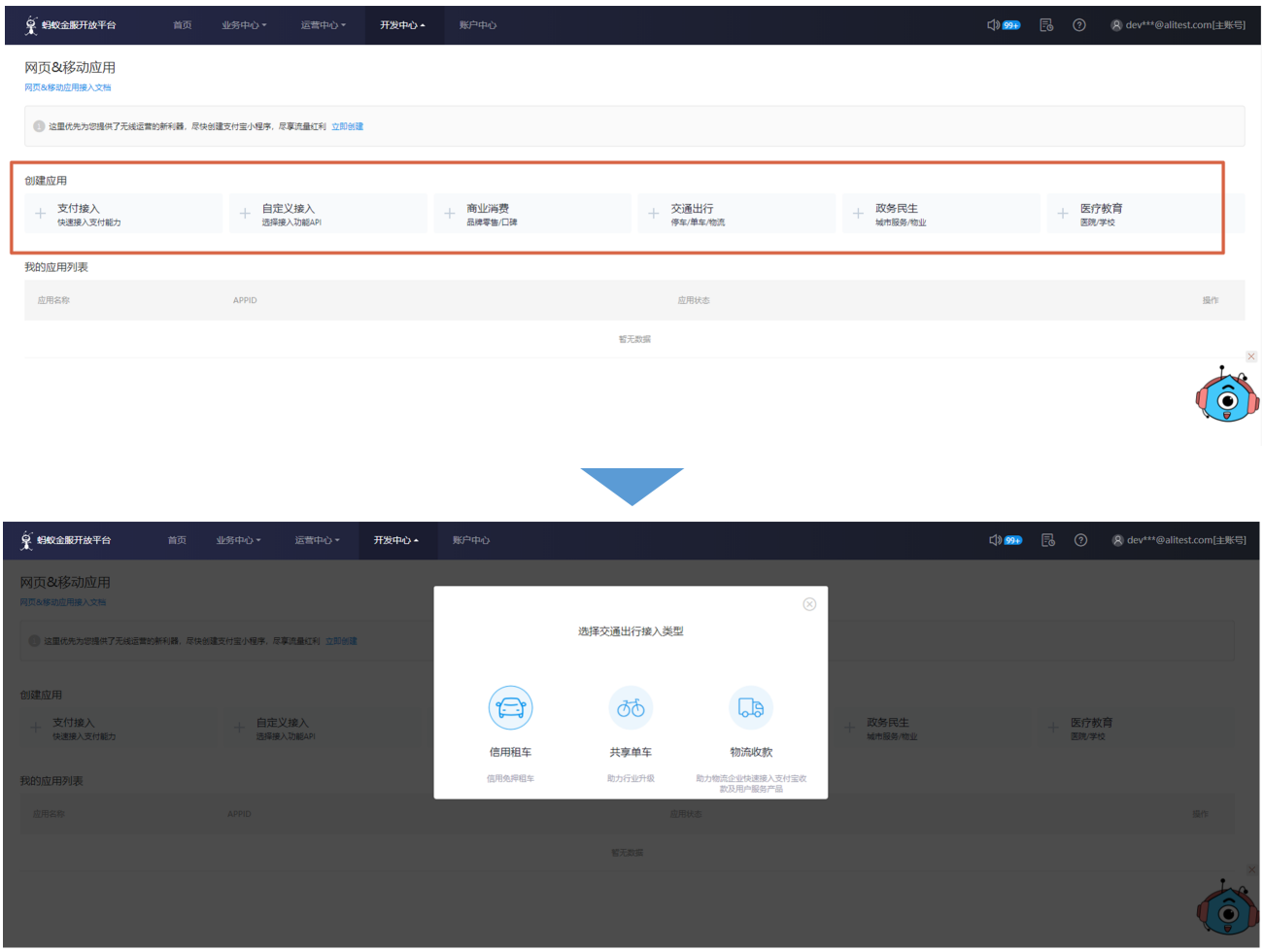
1、注册账号

https://auth.alipay.com/login/index.htm

2、进行企业认证

3、创建应用

https://docs.open.alipay.com/399/106843/


蚂蚁金服开放平台
首页
业务中心
运营中心
开发中心
账户中心
网页&移动应用
创建应用
支付接入
自定义接入
商业消费
交通出行
政务民生
医疗教育
我的应用列表
选择交通出行接入类型
信用租车
共享单车
物流收款

4、开发者配置-获取秘钥


开发配置
使用支付宝的部分功能前，需要先设置应用环境，查看如何使用
基础环境
支付宝网关
https://openapi.alipay.com/gateway.do
应用网关
设置
1
授权回调地址
设置
2
接口加签方式
如何生成密钥?
RSA2(SHA256)密钥(推荐)
设置应用公钥
3
使用其他加签方式
RSA(SHA1)密钥
4

5、完整流程

6、支付宝的相关秘钥

支付宝初体验：

1、依赖

```
<!-- https://mvnrepository.com/artifact/com.alipay.sdk/alipay-sdk-java --> <dependency>
<groupId>com.alipay.sdk</groupId> <artifactId>alipay-sdk-java</artifactId>
<version>3.7.110.ALL</version> </dependency>
```

2、配置Client

``

```
@Configuration
public class AliPayClientConfig {
    @Bean
    public AlipayClient createAC(){
        return new
DefaultAlipayClient(AliPay_Config.PAY_URL,AliPay_Config.APP_ID,AliPay_Config.APP_PRIVATE_KE
Y,"json",AliPay_Config.CHARSET
                ,AliPay_Config.ALIPAY_PUBLIC_KEY,AliPay_Config.SIGN_TYPE);
    }
}
```

``

```
public class AliPay_Config {
    public static final String PAY_URL = "https://openapi.alipay.com/gateway.do";
    public static final String APP_ID = "2017091608770636";
    public static final String APP_PRIVATE_KEY =
"MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCh1qI8uo1qhrcePsa5JUAoYUX8HfPuBt7kc9OaCP
1M/v61/uzaU/lyGQeChKV3jdDTn2Lcq6kT5JBl3TLiaYHmO6cId1nQAIUxiT9zhB9crc4wAx8CVabMbcqUefs7Xsp+Y
hhUgU5X6GOS3emkUeL7RegBnL8vayfEBeUDgBxsk/K/VygBA8sapsEhnoOrB6bhMY4GaJrxb0kg9Ej8x4kpExLcxkT+
UgcOiJvh6vpBZo5CJsiPQkFvSsNsWY2uSDudSL/KqpMxz+yPfVvZDt4fOfyi+CfYR43Jlo4tsT7joqH2JTO6BH+KdJy
c1D3Lqw7w/WdmZtmoLghH0kRZawrLAgMBAAECggEAYYtpm+rhQ7zQ8HTr+DogknYW5Z/0H5qai93d/Uw/yEHFqlJt1i
ZZKlE1upBS311l6beesdzxeuD/u7X4bokjV27K/YpaYsl9fl74FJslAApuRXgMH68aawsd2CIxsBYxPL3JZl3Np6SVJ
7eDlJwakFMRRK+CeIVAoaDf6R01hKctkYnnE0wT+ffQNKWsISoEyiKVT3g5fur7iPOuDlDXsfi6Mm+e75wCXTmRRHmb
8lPBAMLV+Kj5DFxg8dwNz81Fs4ZM2AqOlBaTfy1H1zSlM1m42wcsMYDcgdEH9aq+OgqK+cny6umgs7/Alg7IgV/9b7A
hKdvAqLy2ERUJtooj2QKBgQDeIoDW3HuTq7sBaBnu63f7icT2RM3fApfOiGM4UDtxPvc5dS5S//o3E8p+rbp21FfBey
LOJFd9dg/eu+ETA+63QMPw4Kq4AH/EA5AFohaOQOIKFDjYyxfyD8ajA4USDwdiaW2/vmMeAtGSv+W5zWb9/t49LOTwz
EW904+yOGcmhQKBgQC6guDZOOb4o9nx5XwZXEe2di4MupARHceGzmolyDvs3Qi/w+8QntrDvfqIJoqoxOG5NVi3jtjk
qtJtMaPyxqNWTabWOOTLbrsqlvPUmeCl0j3FVFKAGcV7/b9XkLvh1DtnIe6rhhZCVB4e4bL/katpOTgulhmSMaWIazt
GUOF1DwKBgQCTeobdn/6vuSlsMqhdFppPN1W8ROwDjt4o8iYlwibk9e//hswdsPN3O7zyQ/dzY2FsBIvEHx6zHkpFD6
nMDSVVJzuv1gmiJjqtccwR4V5mTOMuG+TuElCwlkbD/ddAeRfm/6YsOoNN7oMjkiI8LKH/alIOfXT2Zji7YhWaNpZNX
QKBgEU6qOduWS1VdGJrcgLfO+aQOOuSPEN+MD+Dgrb/ee7TpJm5mpUqwbOCWWoMFE/MtJRQjtujdDJ8jZrmYBqPTLWO
IS1G9PXl5idK3Lq/Wzlxrmf+gpj19+2sJEfWeOa5xkrjt3mHTd/U5VFFKXHfmiZ2jLoOEPPI5c6bLudNo/BVAoGBAMv
wRxLO4xb11Ip4rnEHkw3Qn8lrddoC3/m7haHYZ5DyGe8wdCdEi6wyk5MvlNQdqdVg5bqVOAiotIBcd5Pemabun2WaB1
1h/6SSb6wKY4Fnz+H155zaEww4no9BTG9llqQV7H8AS77dN1bxhcpE/MGFoB9JFUOD+BwXAnth4z1u";
    public static final String CHARSET = "UTF-8";
    public static final String ALIPAY_PUBLIC_KEY =
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAx7jJT+PSEM6ZiimTWOSGUfg4cJUO4H/mQqkL2mk7KaHXFQ
qMh4US6xYkDlaEXzOOfxevuBqWOaB4/8TleO1CHZHXWHu9Xc+iYtJPNJGrxoGLM+6Cg9IafJTygRoaqdHOSoVMpxFdO
pUftNdXHO+G0ZpS/7c1zpn8G64zN5J17IFrLcUlsEnSgOrJxsS2Q5Ob44er0KQlj76pehB2sTveHS2vdhqXzrv+oq99
XtUKEY1a3nwDjXneI7YYKLHD9KU53pti/ibLDkOEjO4+DRowd+wfSwkmWGVL3X32OmvCfrg/aMN71B/cyyhWOmQ4cxq
h2UcnpxLmOv/+uC7dSCyAJwIDAQAB";
    public static final String SIGN_TYPE = "RSA2";
}
```

3、实现各种接口的操作

预支付

``

```
    //统一收单线下交易预创建   可以获取支付二维码的链接
    public static String preCreatePay(String payJson) throws AlipayApiException {
      AlipayTradePrecreateRequest request = new AlipayTradePrecreateRequest();
        request.setBizContent(payJson);
        AlipayTradePrecreateResponse response = client.execute(request);
        if(response.isSuccess()){
//            System.out.println("调用成功:"+response.getQrCode());
            return response.getQrCode();
        } else {
//            System.out.println("调用失败");
            return "调用失败:"+response.getMsg();
        }
    }
```

AliPayClient:

在 SDK 调用前需要进行初始化，以 JAVA 代码为例：

```
AlipayClient alipayClient = new
DefaultAlipayClient(URL,APP_ID,APP_PRIVATE_KEY,FORMAT,CHARSET,ALIPAY_PUBLIC_KEY,SIGN_TYPE);
```

关键参数说明：

配置参数	示例值解释	获取方式/示例值
URL	支付宝网关（固定）	https://openapi.alipay.com/gateway.do
APPID	APPID 即创建应用后生成	获取见上方创建应用
APP_PRIVATE_KEY	开发者私钥，由开发者自己生成	获取见配置密钥
FORMAT	参数返回格式，只支持 json	json（固定）
CHARSET	编码集，支持 GBK/UTF-8	开发者根据实际工程编码配置
ALIPAY_PUBLIC_KEY	支付宝公钥，由支付宝生成	获取详见配置密钥
SIGN_TYPE	商户生成签名字符串所使用的签名算法类型，目前支持 RSA2 和 RSA，推荐使用 RSA2	RSA2

接下来，就可以用 alipayClient 来调用具体的 API 了。alipayClient 只需要初始化一次，后续调用不同的 API 都可以使用同一个 alipayClient 对象。

注意： ISV /开发者可以通过第三方应用授权得到商户授权令牌（app_auth_token）作为请求参数传入，实现代商户发起请求的能力。

下单示例代码：

``

```
//统一收单线下交易预创建  可以获取支付二维码的链接
public R preCreatePay(String payJson)  {
  AlipayTradePrecreateRequest request = new AlipayTradePrecreateRequest();
    request.setBizContent(payJson);
    AlipayTradePrecreateResponse response = null;
    try {
        response = client.execute(request);
        if(response.isSuccess()){
            return R.setOK(response.getQrCode(),null);
        } else {
            return R.setERROR("调用失败:"+response.getMsg(),null);
        }
    } catch (AlipayApiException e) {
        e.printStackTrace();
    }
    return R.setERROR("支付宝服务器异常",null);

}
```

1.3 微信支付

https://pay.weixin.qq.com/static/applyment_guide/applyment_index.shtml

xml数据

接入步骤：

1、注册

https://pay.weixin.qq.com/index.php/apply/applyment_home/guide_normal

Native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

2、获取配置

表3.1 账户参数说明

邮件中参数	API参数名	详细说明
APPID	appid	appid是微信公众账号或开放平台APP的唯一标识，在公众平台申请公众账号或者在开放平台申请APP账号后，微信会自动分配对应的appid，用于标识该应用。可在微信公众平台-->开发-->基本配置里面查看，商户的微信支付审核通过邮件中也会包含该字段值。
微信支付商户号	mch_id	商户申请微信支付后，由微信支付分配的商户收款账号。
API密钥	key	交易过程生成签名的密钥，仅保留在商户系统和微信支付后台，不会在网络中传播。商户妥善保管该Key，切勿在网络中传输，不能在其他客户端中存储，保证key不会被泄漏。商户可根据邮件提示登录微信商户平台进行设置。也可按一下路径设置：微信商户平台(pay.weixin.qq.com)-->账户中心-->账户设置-->API安全-->密钥设置
Appsecret	secret	AppSecret是APPID对应的接口密码，用于获取接口调用凭证access_token时使用。在微信支付中，先通过OAuth2.0接口获取用户openid，此openid用于微信内网页支付模式下单接口使用。可登录公众平台-->微信支付，获取AppSecret（需成为开发者且帐号没有异常状态）。

3、微信支付协议

协议规则

商户接入微信支付，调用API必须遵循以下规则：

传输方式	为保证交易安全性，采用HTTPS传输
提交方式	采用POST方法提交
数据格式	提交和返回数据都为XML格式，根节点名为xml
字符编码	统一采用UTF-8字符编码
签名算法	MD5/HMAC-SHA256
签名要求	请求和接收数据均需要校验签名，详细方法请参考安全规范-签名算法
证书要求	调用申请退款、撤销订单、红包接口等需要商户api证书，各api接口文档均有说明。
判断逻辑	先判断协议字段返回，再判断业务返回，最后判断交易状态

验签算法：

签名生成的通用步骤如下：

第一步，设所有发送或者接收到的数据为集合M，将集合M内非空参数值的参数按照参数名ASCII码从小到大排序（字典序），使用URL键值对的格式（即key1=value1&key2=value2...）拼接成字符串stringA。

特别注意以下重要规则：

1. ◆ 参数名ASCII码从小到大排序（字典序）；
2. ◆ 如果参数的值为空不参与签名；
3. ◆ 参数名区分大小写；
4. ◆ 验证调用返回或微信主动通知签名时，传送的sign参数不参与签名，将生成的签名与该sign值作校验。
5. ◆ 微信接口可能增加字段，验证签名时必须支持增加的扩展字段

签名生成算法代码示例：

``

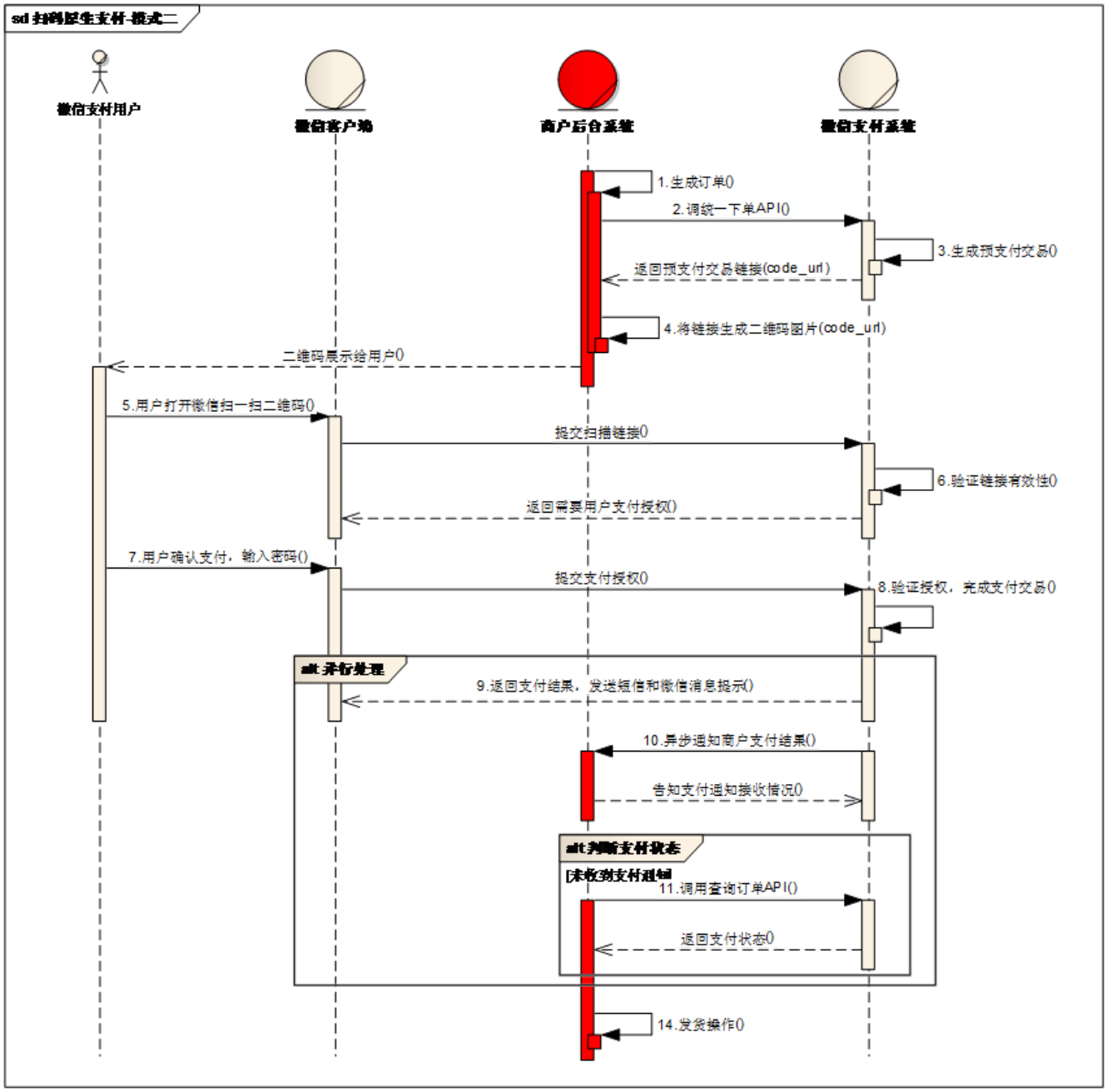
```
/**
 * 生成签名
 */
public static String createSign(String characterEncoding, SortedMap<Object,
        Object> packageParams, String API_KEY) {
    StringBuffer sb = new StringBuffer();
    Set es = packageParams.entrySet();
    Iterator it = es.iterator();
    while (it.hasNext()) {
        Map.Entry entry = (Map.Entry) it.next();
        String k = (String) entry.getKey();
        String v = (String) entry.getValue();
        if (null != v && !"".equals(v) && !"sign".equals(k) &&
                !"key".equals(k)) {
            sb.append(k + "=" + v + "&");
```

```
            }
        }
        sb.append("key=" + API_KEY);
        String sign = MD5Util.MD5Encode(sb.toString(), characterEncoding).toUpperCase();
        return sign;
    }
```

第二步，在stringA最后拼接上key得到stringSignTemp字符串，并对stringSignTemp进行MD5运算，再将得到的字符串所有字符转换为大写，得到sign值signValue。

◆ key设置路径：微信商户平台(pay.weixin.qq.com)-->账户设置-->API安全-->密钥设置

4、微信支付模式



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】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。

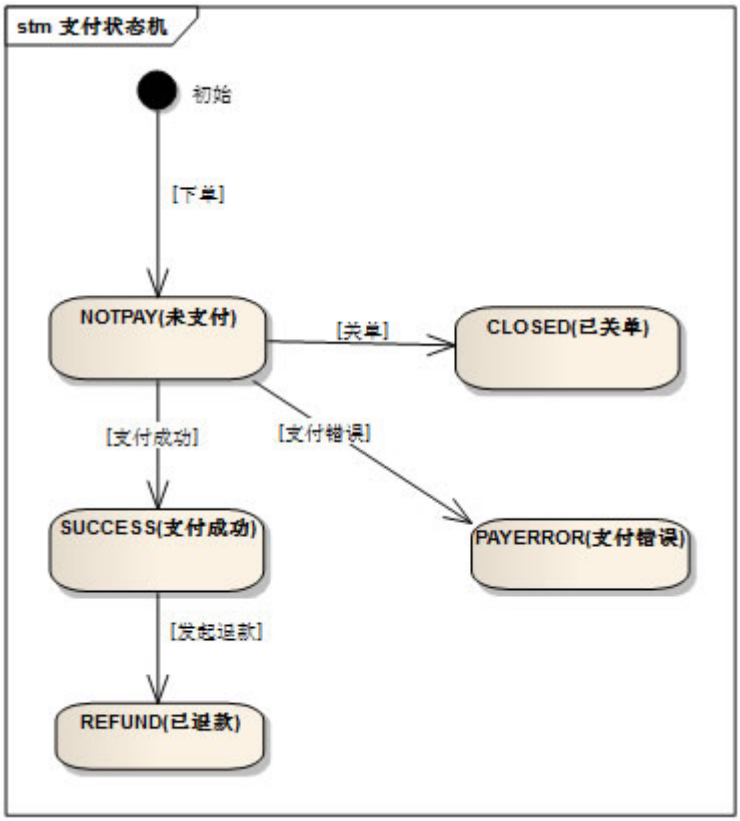
（9）微信支付系统完成支付交易后给微信客户端返回交易结果，并将交易结果通过短信、微信消息提示用户。微信客户端展示支付交易结果页面。

（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】。

（12）商户确认订单已支付后给用户发货。

5、统一下单 生成预支付链接 支付二维码



https://api.mch.weixin.qq.com/pay/unifiedorder

JSAPI--JSAPI支付（或小程序支付）、NATIVE--Native支付、APP--app支付，MWEB--H5支付，不同trade_type决定了调起支付的方式，请根据支付产品正确上传

MICROPAY--付款码支付，付款码支付有单独的支付接口，所以接口不需要上传，该字段在对账单中会出现

代码示例：

``

```
//生成支付数据
/**
 * 下单*/
public static String weixin_pay(WechatPayDto payDto) {
    String ip= null;
    try {
        ip = Inet4Address.getLocalHost().getHostAddress();
        String appid = WechatPayConfig.APP_ID; // appid
        String key = WechatPayConfig.API_KEY; // key
        String currTime = WechatPayCommon.getCurrTime();
        String strTime = currTime.substring(8, currTime.length());
        String strRandom = WechatPayCommon.buildRandom(4) + "";
        String nonce_str = strTime + strRandom;
        //String spbill_create_ip = PayConfig.CREATE_IP; //
        // String notify_url = PayConfig.NOTIFY_URL;
        String trade_type = "NATIVE";//JSAPI--公众号支付、NATIVE--原生扫码支付、APP--app支付，统
一下单接口trade_type的传参可参考这里
        //MICROPAY--刷卡支付，刷卡支付有单独的支付接口，不调用统一下单接口
        SortedMap<Object, Object> packageParams = new TreeMap<Object, Object>();
        packageParams.put("appid", appid);
        packageParams.put("mch_id", WechatPayConfig.MCH_ID);
        packageParams.put("nonce_str", nonce_str);
        packageParams.put("body", payDto.getBody());
        packageParams.put("out_trade_no", payDto.getOut_trade_no());
        // packageParams.put("total_fee", order_price);
        packageParams.put("total_fee", payDto.getOrder_price());
        packageParams.put("spbill_create_ip", ip);
        packageParams.put("notify_url", "http://"+ip+":8080/paycallback");
        packageParams.put("trade_type", trade_type);
        String sign = WechatPayCommon.createSign("UTF-8", packageParams, key);
        packageParams.put("sign", sign);
        String requestXML = WechatPayCommon.getRequestXml(packageParams);
        System.out.println("请求----->"+requestXML);
        String resXml = Http_Util.postData(WechatPayConfig.UFDOOER_URL, requestXML);
        System.out.println("结果----->"+resXml);
        Map map = XmlUtil.doXMLParse(resXml);
        String urlCode = (String) map.get("code_url");
        return urlCode;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;

}
```

1.4 二维码

QrCode

使用步骤:

1、依赖

com.google.zxing

javase

3.3.3

2、编写代码

实现二维码的生成

``

```
public static BufferedImage createQrCode(String msg,int width){
    BufferedImage bufferedImage=null;
    //创建集合 设置二维码的属性
    Map<EncodeHintType,Object> map=new HashMap<>();
    //设置边距
    map.put(EncodeHintType.MARGIN,1);
    //设置图片
    map.put(EncodeHintType.ERROR_CORRECTION,ErrorCorrectionLevel.H);
    //设置内容的编码格式
    map.put(EncodeHintType.CHARACTER_SET,"UTF-8");
    try {
        //
        /**参数说明
         * 1、二维码的内容
         * 2、二维码的类型
         * 3、宽度
         * 4、高度
         * 5、二维码的属性*/
        BitMatrix bitMatrix=new
MultiFormatWriter().encode(msg,BarcodeFormat.QR_CODE,width,width,map);
        MatrixToImageConfig matrixToImageConfig=new
MatrixToImageConfig(0xFF000000,0xFFFFFFFF);
        bufferedImage=MatrixToImageWriter.toBufferedImage(bitMatrix,matrixToImageConfig);

    } catch (WriterException e) {
        e.printStackTrace();
    }
    return bufferedImage;
}
public static String createQrCodeFile(String filePath,String msg,int width){
    BufferedImage bufferedImage=createQrCode(msg, width);
     File file=new File(filePath);
    try {
        ImageIO.write(bufferedImage,"PNG",file);
    } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        return file.getAbsolutePath();
    }
```

1.5 沙箱

沙箱就是测试环境，模拟真实操作，比如支付、IM等

https://docs.open.alipay.com/200/105311

二、支付系统

2.1 设计数据库

2.2 实现支付系统

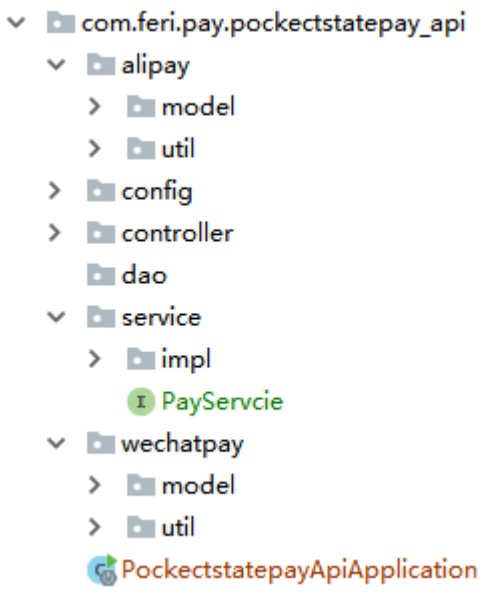
1、依赖

``

```
<!-- https://mvnrepository.com/artifact/com.alipay.sdk/alipay-sdk-java -->
<dependency>
    <groupId>com.alipay.sdk</groupId>
    <artifactId>alipay-sdk-java</artifactId>
    <version>3.7.110.ALL</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.3</version>
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.21</version>
</dependency>
```

2、搭建项目结构



3、代码实现

微信支付工具类:

``

```
@Service
public class AliPayCommon {
    @Autowired
    private AlipayClient client;

    //统一收单线下交易预创建   可以获取支付二维码的链接
    public R preCreatePay(String payJson)  {
      AlipayTradePrecreateRequest request = new AlipayTradePrecreateRequest();
        request.setBizContent(payJson);
        AlipayTradePrecreateResponse response = null;
        try {
            response = client.execute(request);
            if(response.isSuccess()){
                return R.setOK(response.getQrCode(),null);
            } else {
                return R.setERROR("调用失败:"+response.getMsg(),null);
            }
        } catch (AlipayApiException e) {
            e.printStackTrace();
        }
        return R.setERROR("支付宝服务器异常",null);

    }
    //订单支付结果查询
    public R queryPayStatus(String oid){
        AlipayTradeQueryRequest request = new AlipayTradeQueryRequest();
        request.setBizContent("{\"out_trade_no\":\""+oid+"\"}");
        AlipayTradeQueryResponse response = null;
        try {
            response = client.execute(request);
```

```
            if(response.isSuccess()){
                System.out.println("调用成功");
                return R.setOK(response.getTradeStatus(),null);
            } else {
                System.out.println("调用失败");
                return R.setERROR("调用失败:"+response.getMsg(),null);
            }
        } catch (AlipayApiException e) {
            e.printStackTrace();
        }
        return R.setERROR("支付宝服务器异常",null);
    }
    //取消订单支付
    public String closeOrder(String oid){
        AlipayTradeCloseRequest request = new AlipayTradeCloseRequest();
        request.setBizContent("{" +
                "\"trade_no\":\""+oid+"\"}");
        AlipayTradeCloseResponse response = null;
        try {
            response = client.execute(request);
            if(response.isSuccess()){
                System.out.println("调用成功");
                return response.getMsg();
            } else {
                System.out.println("调用失败");
                return response.getMsg();
            }
        } catch (AlipayApiException e) {
            e.printStackTrace();
        }
        return "支付宝服务器异常";
    }
   }
```

支付宝支付工具类:

``

```
public class WechatPayCommon {
    /**
     * 验证签名
     * @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);
        String mysign = MD5Util.MD5Encode(sb.toString(),
                characterEncoding).toLowerCase();
        String tenpaySign = ((String)packageParams.get("sign")).toLowerCase();
        return tenpaySign.equals(mysign);
    }
    /**
     * 生成签名
     */
    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 = (String) entry.getKey();
            String v = (String) entry.getValue();
            if (null != v && !"".equals(v) && !"sign".equals(k) &&
                    !"key".equals(k)) {
                sb.append(k + "=" + v + "&");
            }
        }
        sb.append("key=" + API_KEY);
        String sign = MD5Util.MD5Encode(sb.toString(), characterEncoding).toUpperCase();
        return sign;
    }
    /**
     * 封装请求参数为xml格式的字符串
     */
    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 = (String) entry.getKey();
            String v = (String) entry.getValue();
                if ("attach".equalsIgnoreCase(k) || "body".equalsIgnoreCase(k)) {
                sb.append("<" + k + ">" + "<![CDATA[" + v + "]]></" + k + ">");
            } else {
                sb.append("<" + k + ">" + v + "</" + k + ">");
            } }
        sb.append("</xml>");
        return sb.toString();
    }
    /*获取指定大小的正整数*/
```

```
    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));
    }

    public static String getCurrTime() {
        Date now = new Date();
        SimpleDateFormat outFormat = new SimpleDateFormat("yyyyMMddHHmmss");
        String s = outFormat.format(now);
        return s;
    }
    //生成支付数据
    /**
     * 下单*/
    public static String weixin_pay(WechatPayDto payDto) {
        String ip= null;
        try {
            ip = Inet4Address.getLocalHost().getHostAddress();
            String appid = WechatPayConfig.APP_ID; // appid
            String key = WechatPayConfig.API_KEY; // key
            String currTime = WechatPayCommon.getCurrTime();
            String strTime = currTime.substring(8, currTime.length());
            String strRandom = WechatPayCommon.buildRandom(4) + "";
            String nonce_str = strTime + strRandom;
            //String spbill_create_ip = PayConfig.CREATE_IP; //
            // String notify_url = PayConfig.NOTIFY_URL;
            String trade_type = "NATIVE";//JSAPI--公众号支付、NATIVE--原生扫码支付、APP--app支
付，统一下单接口trade_type的传参可参考这里
            //MICROPAY--刷卡支付，刷卡支付有单独的支付接口，不调用统一下单接口
            SortedMap<Object, Object> packageParams = new TreeMap<Object, Object>();
            packageParams.put("appid", appid);
            packageParams.put("mch_id", WechatPayConfig.MCH_ID);
            packageParams.put("nonce_str", nonce_str);
            packageParams.put("body", payDto.getBody());
            packageParams.put("out_trade_no", payDto.getOut_trade_no());
            // packageParams.put("total_fee", order_price);
            packageParams.put("total_fee", payDto.getOrder_price());
            packageParams.put("spbill_create_ip", ip);
            packageParams.put("notify_url", "http://"+ip+":8080/paycallback");
            packageParams.put("trade_type", trade_type);
            String sign = WechatPayCommon.createSign("UTF-8", packageParams, key);
            packageParams.put("sign", sign);
            String requestXML = WechatPayCommon.getRequestXml(packageParams);
            System.out.println("请求----->"+requestXML);
            String resXml = Http_Util.postData(WechatPayConfig.UFDOOER_URL, requestXML);
            System.out.println("结果----->"+resXml);
```

```
            Map map = XmlUtil.doXMLParse(resXml);
            String urlCode = (String) map.get("code_url");
            return urlCode;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;

    }

    //生成查询xml
    public static String weixin_query(String oid){
        String currTime = getCurrTime();
        String strTime = currTime.substring(8, currTime.length());
        String strRandom = WechatPayCommon.buildRandom(4) + "";
        String nonce_str = strTime + strRandom;
        SortedMap<Object, Object> packageParams = new TreeMap<Object, Object>();
        packageParams.put("appid", WechatPayConfig.APP_ID);
        packageParams.put("mch_id", WechatPayConfig.MCH_ID);
        packageParams.put("out_trade_no", oid);
        packageParams.put("nonce_str", nonce_str);
        String sign = WechatPayCommon.createSign("UTF-8", packageParams,
WechatPayConfig.API_KEY);
        packageParams.put("sign", sign);
        String requestXML = WechatPayCommon.getRequestXml(packageParams);
        System.out.println("查询请求----->"+requestXML);
        String resXml = Http_Util.postData(WechatPayConfig.QUERY_URL, requestXML);
        System.out.println("查询结果----->"+resXml);
        return resXml;
    }
    }
```

2.3 支付接口

微信

支付接口

查询接口

支付宝

支付接口

查询接口

业务接口:

``

```
public interface PayServcie {
    R sendAliPay(AliPayDto payDto);
    R sendWechatPay(WechatPayDto payDto);
    R queryAliPay(String oid);
    R queryWechatPay(String oid);

}
```

业务实现：

``

```
@Service
public class PayServiceImpl implements PayServcie {
    @Autowired
    private AliPayCommon aliPayCommon;
    @Override
    public R sendAliPay(AliPayDto payDto)  {
        R r =aliPayCommon.preCreatePay(JSON.toJSONString(payDto));
        if(r.getCode()==ResultJson_Config.OK){
            return R.setOK(PayConfig.QRCODE_URL+Base64.getUrlEncoder().
                    encodeToString(r.getMsg().getBytes()),null);
        }
        return r;
    }

    @Override
    public R sendWechatPay(WechatPayDto payDto) {
       String r=WechatPayCommon.weixin_pay(payDto);
       if(r!=null){
           return R.setOK(PayConfig.QRCODE_URL+Base64.getUrlEncoder().
                   encodeToString(r.getBytes()),null);
       }else {
           return R.setERROR("微信支付异常，暂无法完成支付",null);
       }
    }

    @Override
    public R queryAliPay(String oid) {
        return aliPayCommon.queryPayStatus(oid);
    }

    @Override
    public R queryWechatPay(String oid) {
        String r=WechatPayCommon.weixin_query(oid);
        Map map=XmlUtil.doXMLParse(r);
        if(map.containsKey("trade_state"))
        {
            String u=map.get("trade_state").toString();
            return R.setOK(u,null);
        }else {
            return R.setERROR("微信支付异常，请稍后再来",null);
        }
```

```
    }
}
```

支付接口：

``

```
@RestController
public class PayController {
    @Autowired
    private PayServcie payServcie;

    @PostMapping("payapi/pay/alipayorder.do")
    public R pay(@RequestBody AliPayDto payDto){
        return payServcie.sendAliPay(payDto);
    }
    @PostMapping("payapi/pay/wechatpayorder.do")
    public R wpay(@RequestBody WechatPayDto payDto){
        return payServcie.sendWechatPay(payDto);
    }
    @GetMapping("payapi/pay/alipayquery/{oid}")
    public R query1(@PathVariable String oid){
        return payServcie.queryAliPay(oid);
    }
    @GetMapping("payapi/pay/wechatpayquery/{oid}")
    public R query2(@PathVariable String oid){
        return payServcie.queryWechatPay(oid);
    }
}
```

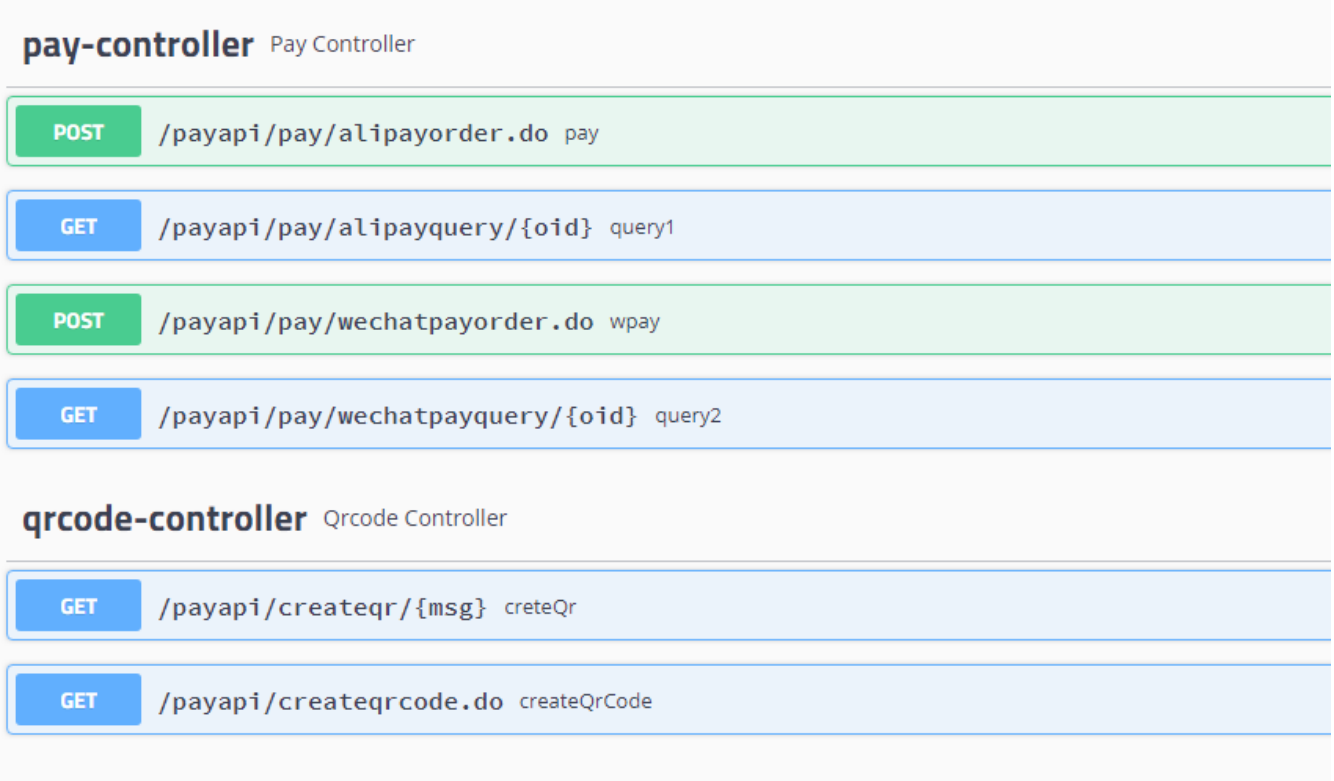
二维码：

生成二维码接口

``

```
@Controller
public class QrcodeController {
    //生成二维码 内容为普通文本
    @GetMapping("/payapi/createqrcode.do")
    public void createQrCode(String msg, HttpServletResponse response) throws IOException {
        BufferedImage bufferedImage=Qrcode_Util.createQrCode(msg,400);
        ImageIO.write(bufferedImage,"png",response.getOutputStream());
    }
    //生成二维码 内容为特殊字符 比如 / : =  编码 base64URL
    @GetMapping("/payapi/createqr/{msg}")
    public void creteQr(@PathVariable String msg,HttpServletResponse response) throws
IOException {
        String m=new String(Base64.getUrlDecoder().decode(msg));
        ImageIO.write(Qrcode_Util.createQrCode(m,400),"png",response.getOutputStream());
    }
}
```

2.4 支付测试



源码地址：https://github.com/xingpenghui/PockectStateApi_Parent/tree/master/PockectStatePay_Api/src/main