支付宝支付——需要使用沙箱测试环境

支付宝支付：

# 沙箱环境地址：

https://openhome.alipay.com/platform/appDaily.htm?tab=info

如果是正式环境:需要用营业执照,申请商户号,appid ， 测试环境就是使用:沙箱环境:https://openhome.alipay.com/platform/appDaily.htm?tab=info
支付宝提供支付接口:给商户使用,来收钱

1.Java,php,C#的 demo,没有 python 的 demo，文件是 pay.py 一般都要自己去写接口
2.git 有人封装了接口和加密算法
3.使用需要安装模块:pip3 install Pycryptodome

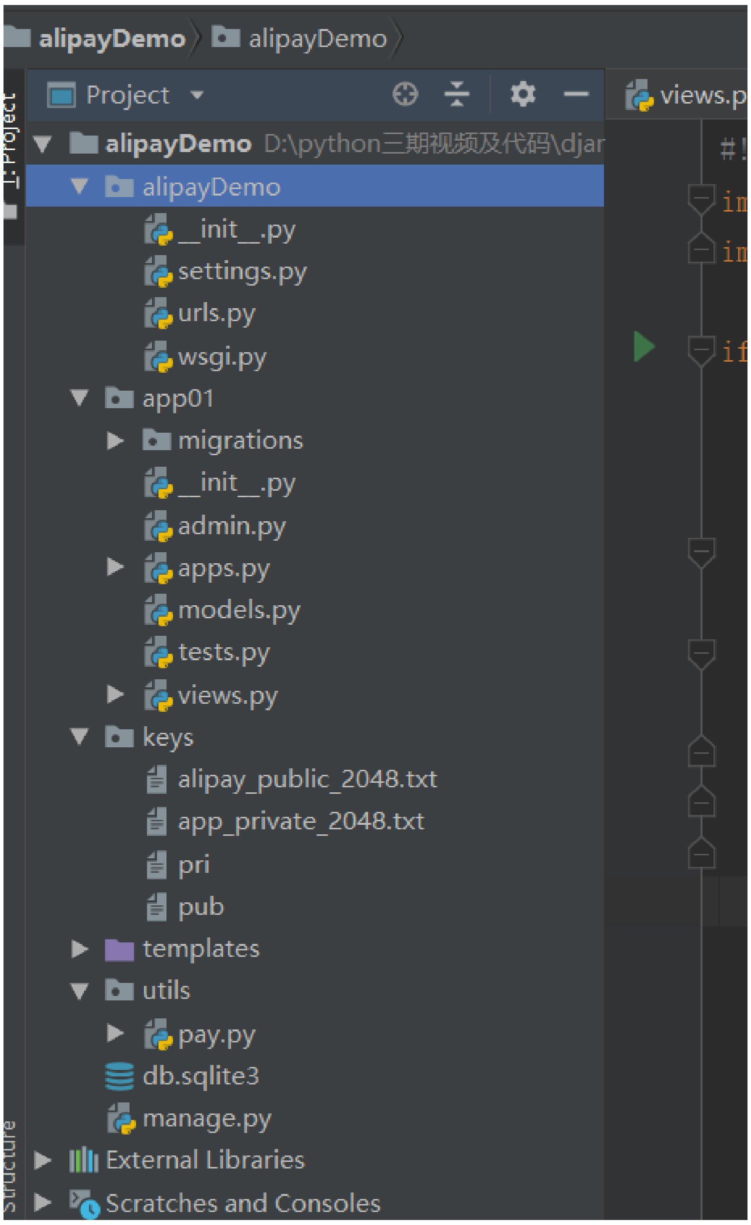
```python
from datetime import datetime
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA256
from urllib.parse import quote_plus
from base64 import decodebytes, encodebytes
import json

class AliPay(object):
    """
    支付宝支付接口(PC 端支付接口)
    """
    def __init__(self, appid, app_notify_url, app_private_key_path,
                 alipay_public_key_path, return_url, debug=False):
        self.appid = appid
        self.app_notify_url = app_notify_url
        self.app_private_key_path = app_private_key_path
        self.app_private_key = None
        self.return_url = return_url
        with open(self.app_private_key_path) as fp:
            self.app_private_key = RSA.importKey(fp.read())
        self.alipay_public_key_path = alipay_public_key_path
        with open(self.alipay_public_key_path) as fp:
            self.alipay_public_key = RSA.importKey(fp.read())
        if debug is True:
            self.__gateway =
"https://openapi.alipaydev.com/gateway.do"
        else:
            self.__gateway = "https://openapi.alipay.com/gateway.do"

    def direct_pay(self, subject, out_trade_no, total_amount,
return_url=None, **kwargs):
        biz_content = {
            "subject": subject,
            "out_trade_no": out_trade_no,
            "total_amount": total_amount,
            "product_code": "FAST_INSTANT_TRADE_PAY",
            # "qr_pay_mode":4
        }

        biz_content.update(kwargs)
        data = self.build_body("alipay.trade.page.pay", biz_content,
self.return_url)
        return self.sign_data(data)

    def build_body(self, method, biz_content, return_url=None):
        data = {
            "app_id": self.appid,
            "method": method,
            "charset": "utf-8",
            "sign_type": "RSA2",
            "timestamp":
datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            "version": "1.0",
            "biz_content": biz_content
        }

        if return_url is not None:
            data["notify_url"] = self.app_notify_url
            data["return_url"] = self.return_url

        return data

    def sign_data(self, data):
        data.pop("sign", None)
        # 排序后的字符串
        unsigned_items = self.ordered_data(data)
        unsigned_string = "&".join("{0}={1}".format(k, v) for k, v in
unsigned_items)
        sign = self.sign(unsigned_string.encode("utf-8"))
        # ordered_items = self.ordered_data(data)
        quoted_string = "&".join("{0}={1}".format(k, quote_plus(v))
for k, v in unsigned_items)

        # 获得最终的订单信息字符串
        signed_string = quoted_string + "&sign=" + quote_plus(sign)
        return signed_string

    def ordered_data(self, data):
        complex_keys = []
        for key, value in data.items():
            if isinstance(value, dict):
                complex_keys.append(key)

        # 将字典类型的数据 dump 出来
        for key in complex_keys:
            data[key] = json.dumps(data[key], separators=(',', ':'))

        return sorted([(k, v) for k, v in data.items()])

    def sign(self, unsigned_string):
        # 开始计算签名
        key = self.app_private_key
        signer = PKCS1_v1_5.new(key)
        signature = signer.sign(SHA256.new(unsigned_string))
        # base64 编码，转换为 unicode 表示并移除回车
        sign = encodebytes(signature).decode("utf8").replace("\n",
"")
        return sign

    def _verify(self, raw_content, signature):
        # 开始计算签名
        key = self.alipay_public_key
        signer = PKCS1_v1_5.new(key)
        digest = SHA256.new()
        digest.update(raw_content.encode("utf8"))
        if signer.verify(digest,
decodebytes(signature.encode("utf8"))):
            return True
        return False
    def verify(self, data, signature):
        if "sign_type" in data:
            sign_type = data.pop("sign_type")
        # 排序后的字符串
        unsigned_items = self.ordered_data(data)
        message = "&".join(u"{}={}".format(k, v) for k, v in
unsigned_items)
        return self._verify(message, signature)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="dist/css/bootstrap.css">
</head>
<body>
    <form method="POST">
        {% csrf_token %}
        <input type="text" name="money">
        <input type="submit" value="去支付" />
    </form>


<script></script>
</body>
</html>
```

```
from django.shortcuts import render, redirect, HttpResponse
from utils.pay import AliPay
import json
import time


def ali():
    # 沙箱环境地址:
https://openhome.alipay.com/platform/appDaily.htm?tab=info
    app_id = "2016092300580099"
    # 支付宝收到用户的支付,会向商户发两个请求,一个 get 请求,一个 post
请求
```

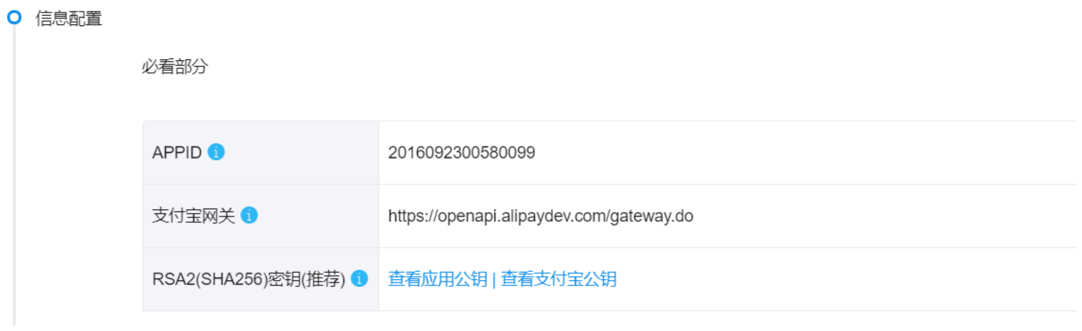
```
    # POST 请求，用于最后的检测
    notify_url = "http://47.52.195.206:80/page2/"
    # notify_url = "http://www.wupeiqi.com:8804/page2/"
    # GET 请求，用于页面的跳转展示
    return_url = "http://47.52.195.206:80/page2/"
    # app 私钥，一定不能丢
    merchant_private_key_path = "keys/app_private_2048.txt"
    # 支付宝公钥
    alipay_public_key_path = "keys/alipay_public_2048.txt"

    # 生成一个 AliPay 的对象
    alipay = AliPay(
        appid=app_id,
        app_notify_url=notify_url,
        return_url=return_url,
        app_private_key_path=merchant_private_key_path,
        alipay_public_key_path=alipay_public_key_path,  # 支付宝的公
钥，验证支付宝回传消息使用，不是你自己的公钥
        debug=True,  # 默认 False,
    )
    return alipay


def page1(request):
    if request.method == "GET":

        return render(request, 'page1.html')
    else:
        money = float(request.POST.get('money'))
        # 生成一个对象
        alipay = ali()
        # 生成支付的 url
        # 对象调用 direct_pay
        query_params = alipay.direct_pay(
            subject="范冰冰版重启娃娃",  # 商品简单描述
            out_trade_no="x2" + str(time.time()),  # 商户订单号
            total_amount=money,  # 交易金额(单位: 元 保留俩位小数)
        )

        pay_url =
"https://openapi.alipaydev.com/gateway.do?{}".format(query_params)
        print(pay_url)
        # 朝这个地址发 get 请求
        return redirect(pay_url)
```

```
def page2(request):
    alipay = ali()
    if request.method == "POST":
        # 检测是否支付成功
        # 去请求体中获取所有返回的数据：状态/订单号
        from urllib.parse import parse_qs
        body_str = request.body.decode('utf-8')
        print(body_str)

        post_data = parse_qs(body_str)
        print(post_data)
        post_dict = {}
        for k, v in post_data.items():
            post_dict[k] = v[0]
        print(post_dict)

        sign = post_dict.pop('sign', None)
        status = alipay.verify(post_dict, sign)
        print('POST 验证', status)
        return HttpResponse('POST 返回')

    else:
        params = request.GET.dict()
        sign = params.pop('sign', None)
        status = alipay.verify(params, sign)
        print('GET 验证', status)
        return HttpResponse('支付成功')
```



```
▼ alipayDemo  D:\python三期视频及代码\djar
  ▼ alipayDemo
      __init__.py
      settings.py
      urls.py
      wsgi.py
  ▼ app01
    ▶ migrations
      __init__.py
      admin.py
    ▶ apps.py
      models.py
      tests.py
    ▶ views.py
  ▼ keys
      alipay_public_2048.txt
      app_private_2048.txt
      pri
      pub
  ▶ templates
  ▼ utils
    ▶ pay.py
    db.sqlite3
    manage.py
▶ External Libraries
▶ Scratches and Consoles
```

4.配置：应用公钥，应用私钥，支付宝公钥



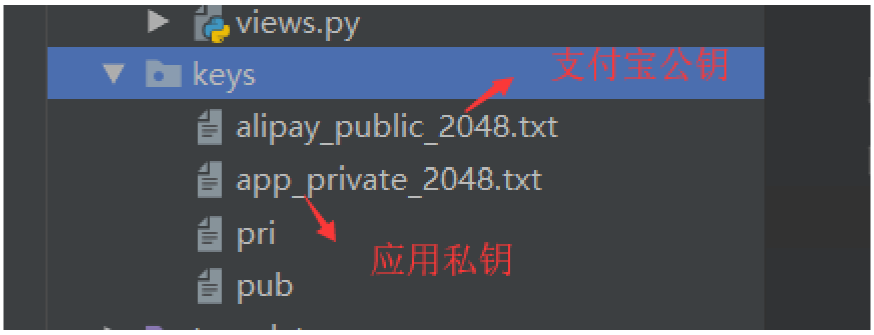
信息配置

必看部分

APPID	2016092300580099
支付宝网关	https://openapi.alipaydev.com/gateway.do
RSA2(SHA256)密钥(推荐)	查看应用公钥 | 查看支付宝公钥

5.应用私钥---自己保存,一定不能丢
6.应用公钥---给别人付款用
7.支付宝公钥---支付宝用的

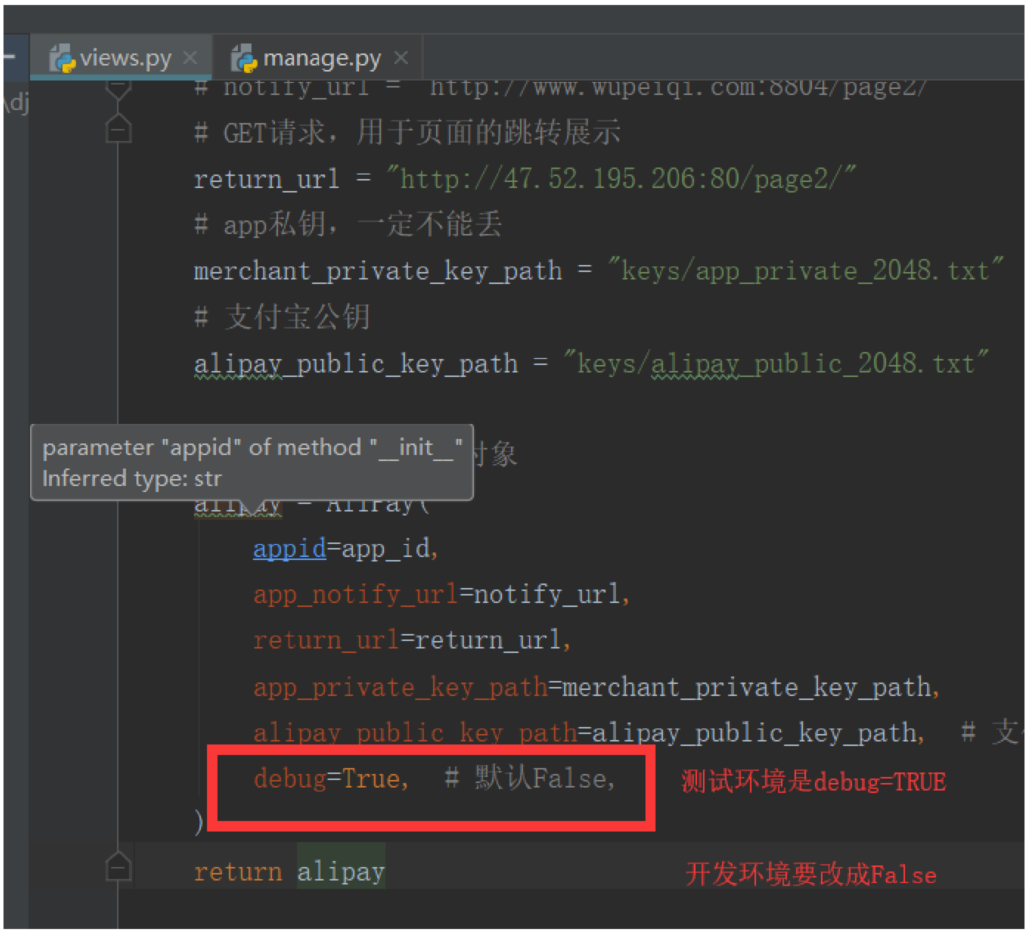
8.生成公钥私钥的地址连接:https://docs.open.alipay.com/291/105971
9.把应用公钥配置在支付宝上:应用公钥,配置完成以后,支付宝自动生成一个支付宝公钥
10.在程序中:配置应用私钥,支付宝公钥


```
▶ views.py
▼ keys
    alipay_public_2048.txt
    app_private_2048.txt
    pri
    pub
```

面试可能会问：如果支付成功,支付宝会回调,但是如果你的服务器挂掉了怎么办?
回答：
支付宝 24 小时以内不定时再给你发,重新启动服务器，你修改掉订单状态即可
支付成功,支付宝会有一个 get 回调,一个 post 回调:修改订单状态

业务逻辑分析：

```
    支付宝支付:
        -demo, 只有 java, php
        -基于支付宝提供的接口
        -git 上有人写好了
        业务逻辑:
        -用户点击去支付按钮, 响应到我们程序， 后台重定向到支付宝支付页面， pay_url =
"https://openapi.alipaydev.com/gateway.do?{}".format(query_params),
页面会显示订单金额和商品信息
          用户扫码支付, 一旦成功, 支付宝会向咱的程序发送两个请求, 一个get 请求, 一个 post 请求
        -通常情况下:get 请求显示页面, post 请求, 通常用来修改订单状态
          需要把项目部署到公网服务器上， 支付宝才可以回调

异步： 把请求放在一个队列里面， 可以使用 redis 实现， 请求来了一个一个排着， 提高网站并发量
```



```
# notify_url = http://www.wupeiqi.com:8804/page2/
# GET请求，用于页面的跳转展示
return_url = "http://47.52.195.206:80/page2/"
# app私钥，一定不能丢
merchant_private_key_path = "keys/app_private_2048.txt"
# 支付宝公钥
alipay_public_key_path = "keys/alipay_public_2048.txt"

parameter "appid" of method "__init__"
Inferred type: str
alipay = AliPay(
    appid=app_id,
    app_notify_url=notify_url,
    return_url=return_url,
    app_private_key_path=merchant_private_key_path,
    alipay_public_key_path=alipay_public_key_path,  # 支
    debug=True,  # 默认False,
)
return alipay
```

测试环境是debug=TRUE

开发环境要改成False