weixin-python Documentation

发布 *0.5.7*

zwczou

2023 年 08 月 06 日

Contents

提供微信登陆，公众号管理，微信支付，微信消息的全套功能

CHAPTER 1

目录

1.1 快速开始

1.1.1 安装

使用 pip

```
sudo pip install weixin-python
```

使用 easy_install

```
sudo easy_install weixin-python
```

安装开发版本

```
sudo pip install git+https://github.com/zwczou/weixin-python@dev
```

1.1.2 功能

- 微信登陆
- 微信支付
- 微信公众号
- 微信消息

1.1.3 用法

异常

父异常类名为 WeixinError 子异常类名分别为 WeixinLoginError WeixinPayError WeixinMPError WeixinMsgError

参数

- WEIXIN_TOKEN 必填，微信主动推送消息的 TOKEN
- WEIXIN_SENDER 选填，微信发送消息的发送者
- WEIXIN_EXPIRES_IN 选填，微信推送消息的有效时间
- WEIXIN_MCH_ID 必填，微信商户 ID，纯数字
- WEIXIN_MCH_KEY 必填，微信商户 KEY
- WEIXIN_NOTIFY_URL 必填，微信回调地址
- WEIXIN_MCH_KEY_FILE 可选，如果需要用退款等需要证书的 api，必选
- WEIXIN_MCH_CERT_FILE 可选
- WEIXIN_APP_ID 必填，微信公众号 appid
- WEIXIN_APP_SECRET 必填，微信公众号 appkey

上面参数的必填都是根据具体开启的功能有关，如果你只需要微信登陆，就只要选择 WEIXIN_APP_ID WEIXIN_APP_SECRET

- 微信消息
  - WEIXIN_TOKEN
  - WEIXIN_SENDER
  - WEIXIN_EXPIRES_IN
- 微信登陆
  - WEIXIN_APP_ID
  - WEIXIN_APP_SECRET
- 微信公众平台
  - WEIXIN_APP_ID
  - WEIXIN_APP_SECRET
- 微信支付
  - WEIXIN_APP_ID

- WEIXIN_MCH_ID
- WEIXIN_MCH_KEY
- WEIXIN_NOTIFY_URL
- WEIXIN_MCH_KEY_FILE
- WEIXIN_MCH_CERT_FILE

初始化

如果使用 flask

```
# -*- coding: utf-8 -*-


from datetime import datetime, timedelta
from flask import Flask, jsonify, request, url_for
from weixin import Weixin, WeixinError


app = Flask(__name__)
app.debug = True


# 具体导入配
# 根据需求导入仅供参考
app.config.fromobject(dict(WEIXIN_APP_ID='', WEIXIN_APP_SECRET=''))


# 初始化微信
weixin = Weixin()
weixin.init_app(app)
# 或者
# weixin = Weixin(app)
```

如果不使用 flask

```
# 根据需求导入仅供参考
config = dict(WEIXIN_APP_ID='', WEIXIN_APP_SECRET='')
weixin = Weixin(config)
```

微信消息

如果使用 django，添加视图函数为

```
url(r'^/$', weixin.django_view_func(), name='index'),
```

如果为 flask，添加视图函数为

```
app.add_url_rule("/", view_func=weixin.view_func)
```

```
@weixin.all
def all(**kwargs):
    """
    监听所有没有更特殊的事件
    """
    return weixin.reply(kwargs['sender'], sender=kwargs['receiver'], content='all')


@weixin.text()
def hello(**kwargs):
    """
    监听所有文本消息
    """
    return "hello too"


@weixin.text("help")
def world(**kwargs):
    """
    监听 help 消息
    """
    return dict(content="hello world!")


@weixin.subscribe
def subscribe(**kwargs):
    """
    监听订阅消息
    """
    print kwargs
    return "欢迎订阅我们的公众号"
```

微信登陆

```
@app.route("/login")
def login():
    """ 登陆跳转地址"""
    openid = request.cookies.get("openid")
    next = request.args.get("next") or request.referrer or "/",
    if openid:
        return redirect(next)

    callback = url_for("authorized", next=next, _external=True)
    url = weixin.authorize(callback, "snsapi_base")
    return redirect(url)


@app.route("/authorized")
def authorized():
    """ 登陆回调函数"""
    code = request.args.get("code")
    if not code:
        return "ERR_INVALID_CODE", 400
    next = request.args.get("next", "/")
    data = weixin.access_token(code)
    openid = data.openid
    resp = redirect(next)
    expires = datetime.now() + timedelta(days=1)
    resp.set_cookie("openid", openid, expires=expires)
    return resp
```

微信支付

注意: 微信网页支付的 timestamp 参数必须为字符串

```
@app.route("/pay/jsapi")
def pay_jsapi():
    """ 微信网页支付请求发起"""
    try:
        out_trade_no = weixin.nonce_str
        raw = weixin.jsapi(openid="openid", body=u"测试", out_trade_no=out_trade_no,␣
↪total_fee=1)
```

(下页继续)

(续上页)

```
        return jsonify(raw)
    except WeixinError, e:
        print e.message
        return e.message, 400


@app.route("/pay/notify")
def pay_notify():
    """
    微信异步通知
    """
    data = weixin.to_dict(request.data)
    if not weixin.check(data):
        return weixin.reply("签名验证失败", False)
    # 处理业务逻辑
    return weixin.reply("OK", True)


if __name__ == '__main__':
    app.run(host="0.0.0.0", port=9900)
```

微信公众号

注意: 如果使用分布式，需要自己实现 access_token 跟 jsapi_ticket 函数

access_token 默认保存在 ~/.access_token jsapi_ticket 默认保存在 ~/.jsapi_ticket

默认在 (HOME) 目录下面，如果需要更改到指定的目录，可以导入库之后修改，如下

```
import weixin

DEFAULT_DIR = "/tmp"
```

获取公众号唯一凭证

```
weixin.access_token
```

获取 ticket

```
weixin.jsapi_ticket
```

创建临时 qrcode

```
data = weixin.qrcode_create(123, 30)
print weixin.qrcode_show(data.ticket)
```

创建永久性 qrcode

```
# scene_id 类型
weixin.qrcode_create_limit(123)
# scene_str 类型
weixin.qrcode_create_limit("456")
```

长链接变短链接

```
weixin.shorturl("http://example.com/test")
```

1.2 微信登陆

1.2.1 功能

- 支持 snsapi_base, 以 snsapi_base 为 scope 发起的网页授权，是用来获取进入页面的用户的 openid 的，并且是静默授权并自动跳转到回调页的。用户感知的就是直接进入了回调页（往往是业务页面）
- 支持 snsapi_userinfo, 以 snsapi_userinfo 为 scope 发起的网页授权，是用来获取用户的基本信息的。但这种授权需要用户手动同意，并且由于用户同意过，所以无须关注，就可在授权后获取该用户的基本信息。

1.2.2 文档

微信详细文档请点击

初始化

```
from weixin import WeixinLogin
# from weixin.login import WeixinLogin

login = WeixinLogin('app_id', 'app_key')
```

引导用户跳转到授权页面

```
url = login.authorize("http://code.show/login/weixin/callback", "snsapi_base")
```

通过 code 换取网页授权 access_token

```
data = login.access_token(code)
print data.access_token
print data.refresh_token
print data.openid
```

小程序通过 js_code 换取 session 跟 openid

```
data = login.jscode2session(code)
```

拉取用户信息

如果是 snsapi_userinfo 则需要

```
login.user_info(data.access_token)
```

刷新 token

```
login.refresh_token(data.refresh_token)
```

1.2.3 用法

```
# -*- coding: utf-8 -*-


from datetime import datetime, timedelta
from flask import Flask, redirect, request, url_for
from weixin.login import WeixinLogin


app = Flask(__name__)

app_id = ''
```

(下页继续)

(续上页)

```
app_secret = ''
wx_login = WeixinLogin(app_id, app_secret)


@app.route("/login")
def login():
    openid = request.cookies.get("openid")
    next = request.args.get("next") or request.referrer or "/",
    if openid:
        return redirect(next)

    callback = url_for("authorized", next=next, _external=True)
    url = wx_login.authorize(callback, "snsapi_base")
    return redirect(url)


@app.route("/authorized")
def authorized():
    code = request.args.get("code")
    if not code:
        return "ERR_INVALID_CODE", 400
    next = request.args.get("next", "/")
    data = wx_login.access_token(code)
    openid = data.openid
    resp = redirect(next)
    expires = datetime.now() + timedelta(days=1)
    resp.set_cookie("openid", openid, expires=expires)
    return resp
```

1.3 微信支付

1.3.1 文档

微信详细文档请点击

初始化

```
from weixin.pay import WeixinPay, WeixinPayError
# 或者
# from weixin import WeixinPay, WeixinError

pay = WeixinPay('app_id', 'mch_id', 'mch_key', 'notify_url', '/path/to/key.pem', '/path/
↪to/cert.pem') # 后两个参数可选
```

统一下单

注意: 默认 `spbill_create_ip` 的值为 `request.remote_addr`，如果没有安装 flask，请在参数后面带上 `spbill_create_ip`

必填参数

- out_trade_no 商户订单号
- body 商品描述
- total_fee 商品费用（分）
- trade_type 交易类型

条件参数

- openid 如果 `trade_type` 为 `JSAPI` 时必须
- product_id 如果 `trade_type` 为 `NATIVE` 时必须

举例

```
try:
    out_trade_no = wx_pay.nonce_str
    raw = pay.unified_order(trade_type="JSAPI", openid="openid", body=u"测试", out_trade_
↪no=out_trade_no, total_fee=1, attach="other info")
    print raw
except WeixinError, e:
    print e.message
```

网页端调起支付 API

必填参数

- out_trade_no 商户订单号
- body 商品描述

- total_fee 商品费用（分）
- openid 用户标识

返回参数

- package 订单详情扩展字符串
- timeStamp 时间戳
- appId 公众号 id
- nonceStr 随机字符串
- signType 清明方式
- sign 签名

```
try:
    out_trade_no = wx_pay.nonce_str
    raw = pay.jsapi(openid="openid", body=u"测试", out_trade_no=out_trade_no, total_
↪fee=1, attach="other info")
    print raw
except WeixinError, e:
    print e
```

查询订单

参数，二选其一

- out_trade_no 商户订单号
- transaction_id 微信订单号

使用 `out_trade_no` 查询订单

```
print pay.order_query(out_trade_no=out_trade_no)
```

或者使用 `transaction_id` 查询

```
print pay.order_query(transaction_id='transaction_id')
```

关闭订单

必填参数

- out_trade_no 商户订单号

举例

```
print pay.order_close(out_trade_no=out_trade_no)
```

申请退款

参数，二选其一，必选填写 key,cert 文件地址

- out_trade_no 商户订单号
- transaction_id 微信订单号

使用 `out_trade_out` 退款

```
print pay.refund(out_trade_no=out_trade_no)
```

或者使用 `transaction_id` 退款

```
print pay.refund(transaction_id='transaction_id')
```

退款查询

参数，4 选一 * out_trade_no 商户订单号 * transaction_id 微信订单号 * out_refund_no 商户退款单号 * refund_id 微信退款单号

使用 `out_trade_no` 退款

```
print pay.refund_query(out_trade_no=out_trade_no)
```

工具函数

签名

```
sign = pay.sign(dict(a='b', b=2, c=3))
```

验证签名

```
pay.check(data(a='b', b=2, c=3, sign=sign))
```

回复消息

```
pay.reply("OK", True)

pay.reply("签名验证失败", False)
```

下载账单

必填参数

- bill_date 账单日期

举例

```
print pay.download_bill('20140603')
```

企业付款

必填参数

- openid 用户身份, amount 金额 (分), partner_trade_no 商户订单号 desc 企业付款备注

举例

```
raw = self.pay.pay_individual(openid=openid, amount=amount, partner_trade_no=partner_
↪trade_no, desc=desc)
print raw
```

查询企业付款

必填参数

- partner_trade_no 商户订单号

举例

```
raw = self.pay.pay_individual_query(partner_trade_no=partner_trade_no)
print raw
```

1.3.2 用法

```
# -*- coding: utf-8 -*-

# from weixin import WeixinPay, WeixinError
from weixin.pay import WeixinPay, WeixinPayError


wx_pay = WeixinPay(app_id, mch_id, mch_key, notify_url)


@app.route("/pay/create")
def pay_create():
    """
    微信 JSAPI 创建统一订单，并且生成参数给 JS 调用
    """
    try:
        out_trade_no = wx_pay.nonce_str
        raw = wx_pay.jsapi(openid="openid", body=u"测试", out_trade_no=out_trade_no,␣
↪total_fee=1)
        return jsonify(raw)
    except WeixinPayError, e:
    # except WeixinError, e
        print e.message
        return e.message, 400


@app.route("/pay/notify", methods=["POST"])
def pay_notify():
    """
    微信异步通知
    """
    data = wx_pay.to_dict(request.data)
    if not wx_pay.check(data):
        return wx_pay.reply("签名验证失败", False)
    # 处理业务逻辑
    return wx_pay.reply("OK", True)


if __name__ == '__main__':
    app.run()
```

1.4 微信消息

1.4.1 来源

本模块主要代码来源于lepture/flask-weixin

1.4.2 功能

- 接收微信推送消息
- 接受微信推送事件
- 发送微信消息

1.4.3 文档

微信详细文档请点击

初始化

```
from weixin import WeixinMsg
# from weixin.msg import WeixinMsg

msg = WeixinMsg('token')

# 如果使用 flask
app.add_url_rule("/", view_func=msg.view_func)

# 如果使用 django
# url(r'^/$', msg.django_view_func(), name='index')
```

接收消息

微信推送消息过来的时候，会先寻找最特殊的规则，然后再是普片的

监听所有

支持的消息类型

- 所有类型 `@msg.all`

- 文本类型 @msg.text() @msg.text("help")
- 图片类型 @msg.image
- 视频类型 @msg.video @msg.shortvideo
- 音频类型 @msg.voice
- 坐标类型 @msg.location
- 链接类型 @msg.link
- 事件类型 @msg.event

```
@msg.all
def all(**kwargs):
    return "所有事件"
```

监听文本消息

```
@msg.text()
def text(**kwargs):
    return "所有文本消息"
```

监听具体的文本消息

```
@msg.text("help")
def help(**kwargs):
    return "可以通过 4001000 联系我们"
```

监听图片消息

```
@msg.image
def image():
    return dict(content="image")
```

监听事件

支持的事件类型

- 订阅事件 @msg.subscribe

- 取消订阅事件 @msg.unsubscribe
- 点击事件 @msg.click
- 其它事件 @msg.{event}

监听订阅事件

```
@msg.subscribe
def subscribe():
    return "欢迎关注我的公众号 code_show"
```

监听取消订阅事件

```
@msg.unsubscribe
def unsubscribe():
    return ""
```

监听点击事件

```
@msg.click
def click(**kwargs):
    print kwargs
    return ""
```

发送消息

支持回复消息的类型

- 文本消息 text
- 音乐消息 music
- 视频消息 video
- 音频消息 voice
- 图片消息 image
- 新闻消息 news

直接在函数里面回复字符串

默认类型为文本消息

```
@msg.click
def click(**kwargs):
    return "欢迎点击"
```

回复字典类型的消息

会自动填充发送者跟接受者

```
@msg.click
def click():
    return dict(content="欢迎点击", type="text")
```

使用 reply 函数

```
@msg.click
def click(**kwargs):
    return msg.reply(kwargs['sender'], sender=kwargs['receiver'], content='click')
```

用法

```
# -*- coding: utf-8 -*-


from flask import Flask
from weixin.msg import WeixinMsg


app = Flask(__name__)
msg = WeixinMsg("e10adc3949ba59abbe56e057f20f883e", None, 0)


app.add_url_rule("/", view_func=msg.view_func)


@msg.all
```

(下页继续)

(续上页)

```
def all_test(**kwargs):
    print kwargs
    # 或者直接返回
    # return "all"
    return msg.reply(
        kwargs['sender'], sender=kwargs['receiver'], content='all'
    )


@msg.text()
def hello(**kwargs):
    return dict(content="hello too!", type="text")


@msg.text("world")
def world(**kwargs):
    return msg.reply(
        kwargs['sender'], sender=kwargs['receiver'], content='hello world!'
    )


@msg.image
def image(**kwargs):
    print kwargs
    return ""


@msg.subscribe
def subscribe(**kwargs):
    print kwargs
    return ""


@msg.unsubscribe
def unsubscribe(**kwargs):
    print kwargs
    return ""


if __name__ == '__main__':
```

(下页继续)

(续上页)

```
    app.run(host="0.0.0.0", port=9900)
```

1.5 微信公众平台

1.5.1 文档

初始化

```
from weixin.mp import WeixinMP

mp = WeixinMP(app_id, app_secret)
```

获取公众号唯一凭证

```
print mp.access_token
```

获取 jsticket

```
print mp.jsapi_ticket
```

jsapi 签名，比如扫码或者分享

```
mp.jsapi_sign(url=request.url)
```

创建临时 qrcode

```
data = mp.qrcode_create(123, 30)
print mp.qrcode_show(data.ticket)
```

创建永久性 qrcode

```
# scene_id 类型
mp.qrcode_create_limit(123)
```

(下页继续)

(续上页)

```
# scene_str 类型
mp.qrcode_create_limit("456")
```

长链接变短链接

```
mp.shorturl("http://example.com/test")
```

菜单管理

```
# 获取菜单
try:
    print mp.menu_get()
except WeixinError:
    pass

# 创建菜单
data = [
    {
        "type": "view",
        "name": "测试",
        "url": "http://code.show/",
    },
]
print mp.menu_create(data)

# 删除菜单
print mp.menu_delete()

# 模板消息
print mp.get_all_private_template()
print mp.del_private_template("oHmefUCu3hUa1r23iun2gP3BM9MVn11g7Ob2J4VzpOg")

data = {
    "first": {
        "value":u"恭喜你购买成功！",
        "color":"#173177"
    },
    "accountType":{
```

(下页继续)

(续上页)

```
        "value":u"巧克力",
        "color":"#173177"
    },
    "account": {
        "value":u"39.8 元",
        "color":"#173177"
    },
    "amount": {
        "value":u"2014 年 9 月 22 日",
        "color":"#173177"
    },
    "result": {
        "value":u"2014 年 9 月 22 日",
        "color":"#173177"
    },
    "remark":{
        "value":u"欢迎再次购买\nabc\nefg",
        "color":"#173177"
    }
}

print mp.template_send("oHmefUCu3hUa1r23iun2gP3BM9MVn11g7Ob2J4VzpOg",
↪"oYhHdsswUDolWKEbeybuAOsHr5W4", data)
```

更多用法参考 example/mp.py

TODO

- [X] 自定义菜单
- [X] 用户管理
  - [X] 用户分组管理
  - [X] 设置用户备注名
  - [X] 获取用户基本信息
  - [X] 获取用户列表
  - [X] 获取用户地理位置
- [X] 账号管理
  - [X] 生成带参数的二维码

  - [X] 长链接转短链接
  - [X] 微信认证事件推送
- [X] 消息管理
  - [X] 普通消息 微信消息
  - [X] 模板消息
- [ ] 素材管理