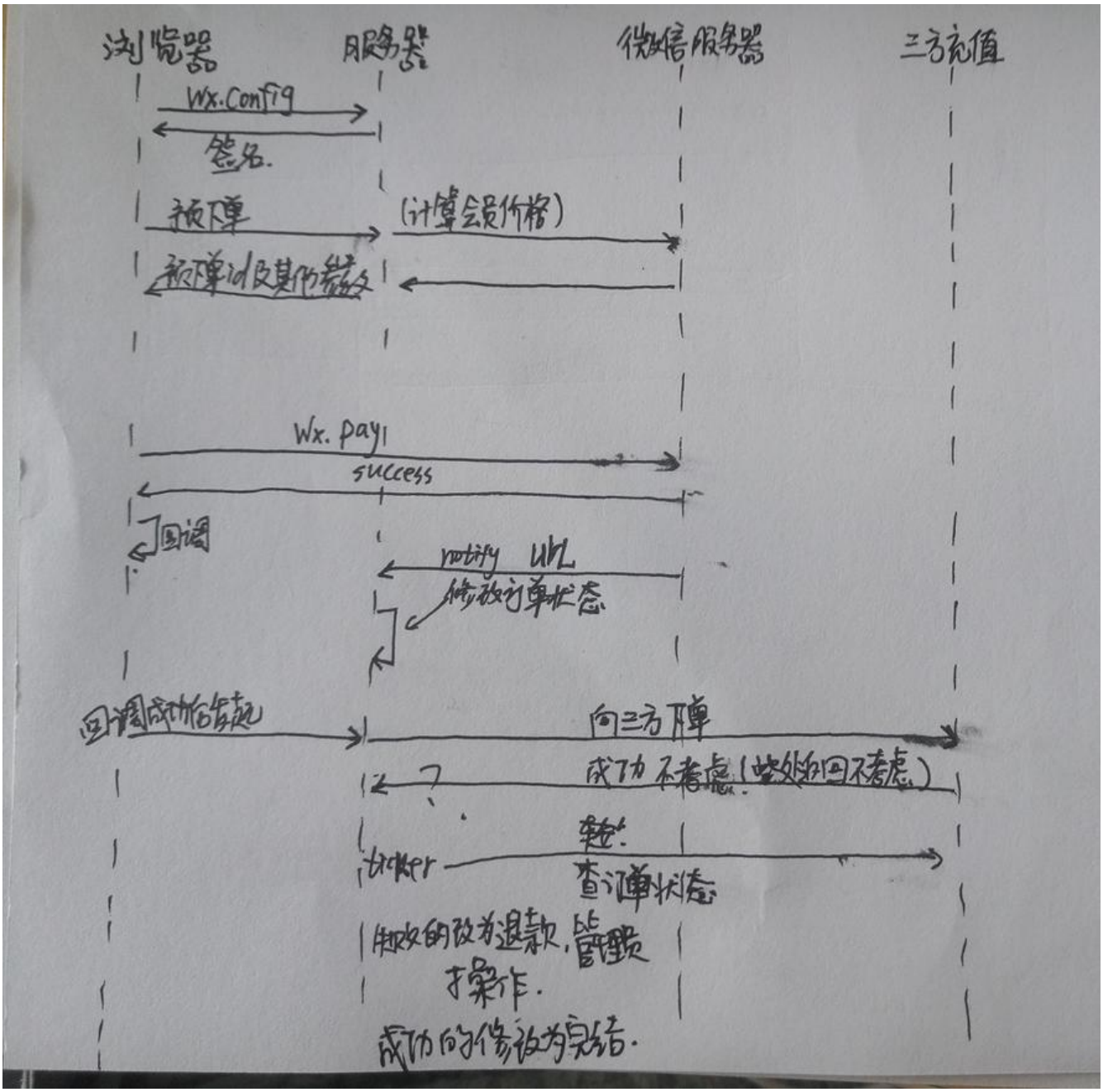
# reactjs +golang 微信支付坑坑坑

作者：xhaoxiong
原文链接：https://ld246.com/article/1539882113984
来源网站：链滴




最近在做支付充值这一块，用的微信支付，发现坑点真不少。由于无golang的sdk于是拿着github上别人写好的改了改，最终也还是圆满完成了支付这一块。

主要场景:微信支付-H5-微信内发起支付

主要内容:微信网页开发>jssdk>微信商户

开发语言:reactjs+golang

大概的主体思路如下每一步的操作都很重要

微信网页开发配置:

1、在微信后台配置域名(目的:获取openid)

2、jssdk域名的配置(目的:前端请求的配置)

3、商户后台的域名配置（目的:支付的配置）

主要配置就以上这些（应该没遗漏）

主要文档：网页授权的文档，jssdk wxconfig wxpay的文档，微信商户支付文档

坑1、微信授权 redrect_uri错误（此处偷懒可以用别人的<a href="gopkg.in/chanxuehong/wechat v2">包</a>）：自己写则仔细检查各种参数以及微信后台的配置即可，总会成功的

示例授权代码:

```
package controllers

import (
    "net/url"

    "charge/models"

    "github.com/astaxie/beego"
    mpoauth2 "gopkg.in/chanxuehong/wechat.v2/mp/oauth2"
    oauth "gopkg.in/chanxuehong/wechat.v2/oauth2"
    "gopkg.in/chanxuehong/wechat.v2/open/oauth2"
)

type Common struct {
    beego.Controller
    UserID uint
}


type WebController struct {
    Common
}

func (this *WebController) Prepare() {
    if !UserInterceptor(&this.Common) {
        WechatLogin(&this.Common)
    }
}

var oauth2Endpoint oauth.Endpoint = mpoauth2.NewEndpoint(Appid, AppSecret)

func WechatLogin(this *Common) {

    code := this.GetString("code")

    //第一次请求进来时候
    if code == "" {
        this.GoAuth()
        this.StopRun()
        return
    }
    client := oauth.Client{Endpoint: oauth2Endpoint}

    if token, err := client.ExchangeToken(code); err != nil {
        beego.Warning("exchange token error:", err)
        this.GoAuth()
```

```
        this.StopRun()
        return
    } else {
        if info, err := oauth2.GetUserInfo(token.AccessToken, token.OpenId, "", nil); err != nil {
            beego.Warning("get userinfo error:", err)
            this.GoAuth()
            this.StopRun()
            return
        } else {
            //保存用户信息到数据库
            var user models.User
            if models.DB.Where("openid =?", info.OpenId).First(&user).RecordNotFound() {

                user.Openid = info.OpenId
                user.Nickname = info.Nickname
                if err := models.DB.Create(&user).Error; err != nil {
                    beego.Error("update user error:", err)
                    this.StopRun()
                    return
                }
                this.UserID = user.ID
                this.SetSession("userinfo", user)
            } else {
                if err := models.DB.Model(&user).Updates(map[string]interface{}{"Nickname": info.N
ckname,}).Error; err != nil {
                    beego.Error("user update error:", err)
                    this.StopRun()
                    return
                }
                this.UserID = user.ID
                this.SetSession("userinfo", user)
            }
        }

    }
}

func (this *Common) GoAuth() {
    //此处values
    uri, _ := url.Parse(beego.AppConfig.String("domain") + this.Ctx.Input.URI())

    values := uri.Query()
    values.Del("code")
    path := mpoauth2.AuthCodeURL(Appid, beego.AppConfig.String("domain")+
        this.Ctx.Input.URL()+
        "?"+ values.Encode(),
        "snsapi_userinfo", this.GetString("token"))
    this.Redirect(path, 302)
    this.StopRun()
    return
}

func UserInterceptor(ctr *Common) bool {
    if userinfo := ctr.GetSession("userinfo"); userinfo != nil {
```

```
        user := userinfo.(models.User)
        if err := models.DB.Where("id=? ", user.ID).First(&user).Error; err != nil {
            beego.Warning("user read error:", err)
            return false
        }

        ctr.UserID = user.ID
        return true
    }

    return false

}

//@router /* [*]
func (this *WebController) Index() {

    this.TplName = "react/index.html"
}

//@router /MP_verify_zpcrQi7YFTe5pSaV.txt [*]
func (this *WebController) MP() {
    this.TplName = "MP_verify_zpcrQi7YFTe5pSaV.html"
}
```

1、请求 ------>服务器---302----->微信服务器（用户同意）

2、微信服务器--code-->服务器

3、服务器----code->微信服务器----(用户信息和token)------服务器

坑2、jssdk配置wx.config 和wx.chooseWXPay

reactjs的微信jssdk　weixin-js-sdk

config配置:react获取当前路径我是使用window然后split域名，去掉默认的#号是使用browserHistor 而不是用hash路由 我的每次wx.chooseWXPay请求之前都是调用了wx.config config

关于调试:需线上配置好的环境在微信开发者工具中调试

关于签名失败:仔细检查参数

[！wx.config基于微信后台域名的配置]

前端代码：

```
_getInitialState() {
        let url = window.location.href;
        let urlArr = url.split("xxxx");
        let that = this;
        let pathname = urlArr[1];
        window.$http.post('/api/jssdk', qs.stringify({path: pathname})).then(res => {
            if (res.status === 10000) {
                that.setState({
```

```
                openid: res.openid
            });
            wx.config({
                debug: true, // 开启调试模式,调用的所有api的返回值会在客户端alert出来，若要查看入的参数，可以在pc端打开，参数信息会通过log打出，仅在pc端时才会打印。
                appId: res.appid, // 必填，公众号的唯一标识
                timestamp: res.timestamp, // 必填，生成签名的时间戳
                nonceStr: res.noncestr, // 必填，生成签名的随机串
                signature: res.signature,// 必填，签名
                jsApiList: ["chooseWXPay", "onMenuShareTimeline", "addCard"], // 必填，需要使的JS接口列表
                success: function (res) {
                }


            });
        }
    }).catch(err => {
    });

}
```

后端jssdk接口

```
//@router /api/jssdk [*]
func (this *WebController) JSSDK() {

    path := this.GetString("path")


    noncestr := string(utils.Krand(16, 3))
    timestamp := time.Now().Unix()
    var user models.User

    //由于已经微信授权将openid写入数据库了
    if err := models.DB.Where("id = ?", this.UserID).First(&user); err != nil {
        beego.Warning(err)
    }
    //"gopkg.in/chanxuehong/wechat.v2/mp/jssdk"
    signature := jssdk.WXConfigSign(JSTICKET, noncestr, strconv.FormatInt(timestamp, 10), 你域名+path)


    data := make(map[string]interface{})
    data["appid"] = Appid
    data["noncestr"] = noncestr
    data["timestamp"] = timestamp
    data["signature"] = signature
    data["status"] = 10000
    data["openid"] = user.Openid
    this.Data["json"] = data
    this.ServeJSON()
    return
```

```
}
```

然后预下单

将openid 商品信息提交到服务端,服务端通过预下单接口下单（此处其实是模拟商户向微信请求此时户的的参数包括）

```
//向商户下单[github.com/objcoding/wxpay]
func PostPreOrder(openid string, tradeNo string, amount int64, ip string, category int) (prepay
d string, paySign string, isOk bool, err error) {
    account := wxpay.NewAccount(Appid, Mchid, ApiKey, false)
    client := wxpay.NewClient(account)

    bodyString := ""

    if category == 1 {
        bodyString = "callpay"
    } else {
        bodyString = "fuelpay"
    }
    // 设置http请求超时时间
    client.SetHttpConnectTimeoutMs(2000)

    // 设置http读取信息流超时时间
    client.SetHttpReadTimeoutMs(1000)
    client.SetSignType("MD5")
    params := make(wxpay.Params)

    fmt.Println("body:", bodyString)
    fmt.Println("appid:", Appid)
    fmt.Println("out_trade_no:", tradeNo)

    params.SetString("body", bodyString).
        SetString("appid", Appid).
        SetString("out_trade_no", tradeNo).
        SetInt64("total_fee", amount).
        SetString("spbill_create_ip", ip).
        SetString("notify_url", notifyUrl).
        SetString("trade_type", tradeType).
        SetString("openid", openid)
    /*
        beego.Error("-----------request param------------")

        for key, value := range params {
            beego.Error("key:", key, " value:", value)
        }
    */
    returnParams, err := client.UnifiedOrder(params)

    if err != nil {
        log.Println(err)
        return "", "", false, err
    }
    /*
    beego.Error("-----------response param------------")
```

```
    for key, value := range returnParams {
        beego.Error("key:", key, " value:", value)
    }
    */
    returnCode, ok := returnParams["return_code"]
    resultCode, ok := returnParams["result_code"]
    if returnCode == "SUCCESS" && resultCode == "SUCCESS" && ok {
        paySign, _ = returnParams["sign"]
        prepayId, _ := returnParams["prepay_id"]
        return prepayId, paySign, true, err
    }
    return "", "", false, err
}
```

上面返回了预下单的preid 和paysign（这是错误的签名） 后面我拿出来重新改写了

```
func (this *OrderController)  test(){
    noncestr := utils.Str2Md5(time.Now().Format("20060102150405"))
    timestamp := time.Now().Unix()

    //来自上面PostPreOrder()返回的
    paySign = Sign(prepayId, noncestr, strconv.FormatInt(timestamp, 10))
    /*

    rst["package"] = "prepay_id=" + prepayId
    rst["paySign"] = paySign
    rst["nonceStr"] = noncestr
    rst["timeStamp"] = strconv.FormatInt(timestamp, 10)
    rst["appId"] = Appid
    rst["status"] = 10000
    rst["tradeNo"] = tradeNo
    this.Data["json"] = rst
    this.ServeJSON()

}

/支付的签名函数 (包内的签名函数的改写)
func Sign(prepayId string, noncestr string, timestamp string) string {
    params := make(wxpay.Params)
    params.SetString("package", "prepay_id="+prepayId).
        SetString("nonceStr", noncestr).
        SetString("timeStamp", timestamp).
        SetString("appId", Appid).
        SetString("signType", "MD5")
    var keys = make([]string, 0, len(params))
    for k := range params {
        if k != "sign" { // 排除sign字段
            keys = append(keys, k)
        }
    }

    sort.Strings(keys)
```

```
    //创建字符缓冲
    var buf bytes.Buffer
    for _, k := range keys {
        if len(params.GetString(k)) > 0 {
            buf.WriteString(k)
            buf.WriteString(`=`)
            buf.WriteString(params.GetString(k))
            buf.WriteString(`&`)
        }
    }
    // 加入apiKey作加密密钥
    buf.WriteString(`key=`)
    buf.WriteString(ApiKey)

    var (
        dataMd5 [16]byte
        str     string
    )
    dataMd5 = md5.Sum(buf.Bytes())
    str = hex.EncodeToString(dataMd5[:])
    return strings.ToUpper(str)
}
```

返回给前端6个重要参数 其中5个是调用wx.chooseWXPay的(这是后端给前端的，也有一些写法是前把参数准备好然后再请求)

此处注意参数key和value的准确性(比如大小写,值的类型等)

在以上基础上再调用wx.chooseWXPay

```
_preOrder = () => {

        let that = this;
        let itemId = that.state.itemId;
        let mobile = that.state.mobile;
        let openid = that.state.openid;
        let category = that.state.category;

        mobile = mobile.replace(/\s/ig, '');
        //预下单
        window.$http.post('/api/order/pre', qs.stringify({
            mobile: mobile,
            id: itemId,
            category: category,
            openid: openid
        })).then(res => {
            //回调获得支付参数(也有是在前端将参数准备好的)
            let appId = res.appId;
            let timeStamp = res.timeStamp;
            let nonceStr = res.nonceStr;
            let packages = res.package;
            let paySign = res.paySign;
            that.setState({
                tradeNo: res.tradeNo
```

```
        });
        //发起支付[此处注意这个包帮你封装了一层 timestamp 实际微信文档为timeStamp]
        wx.chooseWXPay({
            appId: appId,
            timestamp: timeStamp.toString(),
            nonceStr: nonceStr,
            package: packages,
            signType: 'MD5',
            paySign: paySign,
            success: function (res) {
                //完成后修改订单状态
                that._paySuccess();
            },
        }).catch(err => {
            console.log("error")
        });

        if (res.status === 10001) {
            console.log("下单失败")
        }
    }).catch(err => {

    });
};
```

总结：整个逻辑很简单但是一个人前后端都写的话，对于微信这一块不熟的话写起来就会犯迷糊，忘了参数，类型等等等。相比来说支付宝的就简单很多，加油