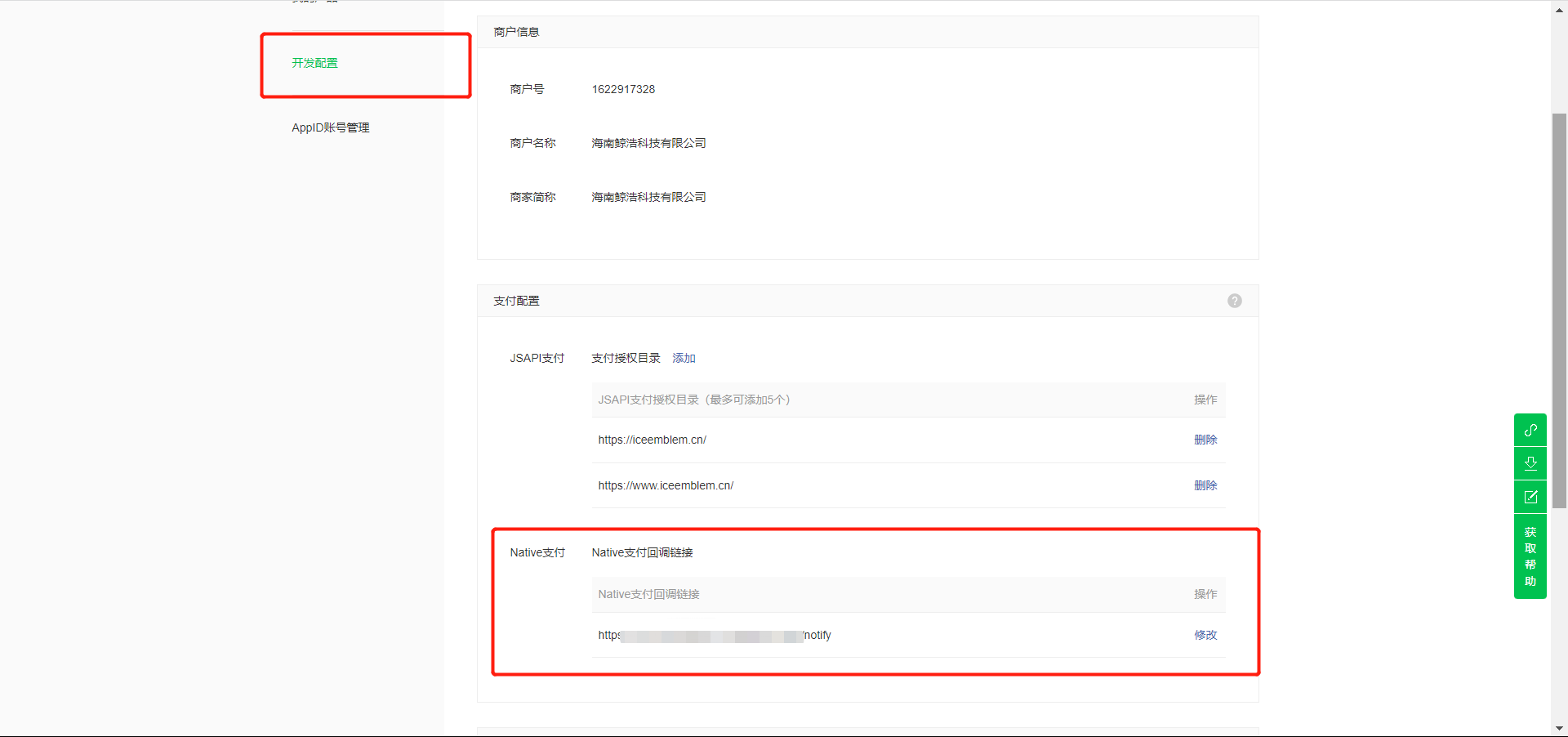
以下文档基于nuget包SKIT.FlurlHttpClient.Wechat

**微信支付准备工作**

首先先做好微信支付准备工作的文档内容，然后需要我们开通Native支付



然后需要配置我们的Native回调地址



**支付帮助类**

首先创建一个支付帮助类

public class WxPayHelper: ITransientDependency

{

    /// <summary>

    /// 平台证书，用于验证微信发送过来的数据

    /// </summary>

    static List<CertificateEntry> \_CertificateEntry { get; set; } = new List<CertificateEntry>();

    /// <summary>

    /// 刷新腾讯支付证书，需要每隔12个小时调用一次

    /// </summary>

    /// <returns></returns>

    public static async Task FetchPayCertificate()

    {

        const string ALGORITHM\_TYPE = "RSA";

        var manager = new InMemoryCertificateManager();

        var options = new WechatTenpayClientOptions()

        {

            MerchantId = WebConfiguration.WxConfig.MerchantId,

            MerchantV3Secret = WebConfiguration.WxConfig.SecretV3,

            MerchantCertificateSerialNumber = WebConfiguration.WxConfig.CertSerialNumber,

            MerchantCertificatePrivateKey = WebConfiguration.WxConfig.CertPrivateKey,

            PlatformCertificateManager = manager // 证书管理器的具体用法请参阅下文的高级技巧与加密、验签有关的章节

        };

        var WxClient = new WechatTenpayClient(options);

        var response = await WxClient.ExecuteQueryCertificatesAsync(new QueryCertificatesRequest() {

            AlgorithmType = ALGORITHM\_TYPE

        });

        if (response.IsSuccessful())

        {

            // 解密响应

            response = WxClient.DecryptResponseSensitiveProperty(response);

            // 存放证书

            \_CertificateEntry = new List<CertificateEntry>();

            foreach (var certificate in response.CertificateList)

            {

                \_CertificateEntry.Add(new CertificateEntry(ALGORITHM\_TYPE, certificate));

            }

        }

}

    /// <summary>

    /// 微信客户端

    /// </summary>

    /// <returns></returns>

    public WechatTenpayClient GetWechatTenpayClient() {

        var manager = new InMemoryCertificateManager();

        foreach (var item in \_CertificateEntry) {

            manager.AddEntry(item);

        }

        var options = new WechatTenpayClientOptions()

        {

            MerchantId = WebConfiguration.WxConfig.MerchantId,

            MerchantV3Secret = WebConfiguration.WxConfig.SecretV3,

            MerchantCertificateSerialNumber = WebConfiguration.WxConfig.CertSerialNumber,

            MerchantCertificatePrivateKey = WebConfiguration.WxConfig.CertPrivateKey,

            PlatformCertificateManager = manager // 证书管理器的具体用法请参阅下文的高级技巧与加密、验签有关的章节

        };

        return new WechatTenpayClient(options);

    }

}

**下单**

/// <summary>

/// 下单

/// </summary>

/// <param name="input"></param>

/// <returns></returns>

/// <exception cref="UserFriendlyException"></exception>

[HttpPost]

public async Task<PayOrderDto> RechargeAsync(RechargeInput input) {

    // 充值金额

    var price = Math.Round(input.Price, 2);

    string description = $"账号充值{price}元";

    // 订单号

    string orderNumber = Tool.CommonOrderNumberCreate();

    // 执行下单动作

    WechatTenpayClient wxClient = WxPayHelper.GetWechatTenpayClient();

    var respone = await wxClient.ExecuteCreatePayTransactionNativeAsync(new SKIT.FlurlHttpClient.Wechat.TenpayV3.Models.CreatePayTransactionNativeRequest() {

        AppId = WebConfiguration.WxConfig.AppId,

        Description = description,

        OutTradeNumber = orderNumber,

        NotifyUrl = WebConfiguration.WxConfig.NotifyUrl,

        Amount = new SKIT.FlurlHttpClient.Wechat.TenpayV3.Models.CreatePayTransactionNativeRequest.Types.Amount() {

            Total = (int) (price \* 100),

            Currency = "CNY"

        }

});

    if (!respone.IsSuccessful()) {

        Logger.LogError($"微信下单失败,订单号:{orderNumber}，错误消息：{respone.ErrorMessage}，错误码：{respone.ErrorCode}");

        throw new UserFriendlyException($"下单失败，错误消息：{respone.ErrorMessage}，错误码：{respone.ErrorCode}");

    }

    Logger.LogInformation($"微信下单成功,订单号:{orderNumber}");

    // 返回支付url，前端拿到url后，根据url生成二维码，用户使用微信扫描完成支付

    return respone.QrcodeUrl;

}

**接收微信通知**

用户支付成功后微信会通知我们，我们需要创建一个api来接收微信通知

我们的controller代码如下

/// <summary>

/// 微信通知接口, 给微信使用的

/// </summary>

/// <returns></returns>

[HttpPost]

[ActionName("notify")]

public async Task<object> Notify(

    [FromHeader(Name = "Wechatpay-Timestamp")] string timestamp,

    [FromHeader(Name = "Wechatpay-Nonce")] string nonce,

    [FromHeader(Name = "Wechatpay-Signature")] string signature,

    [FromHeader(Name = "Wechatpay-Serial")] string serialNumber)

{

    using var reader = new StreamReader(Request.Body, Encoding.UTF8);

    string body = await reader.ReadToEndAsync();

    // 调用业务层方法

    var success = await WxPayAppService.Notify(timestamp, nonce, signature, serialNumber, body);

    return success ? new JsonResult(new { code = "SUCCESS", message = "成功" }) : new JsonResult(new { code = "FAIL", message = "失败" });

}

我们的WxPayAppService.Notify代码如下

/// <summary>

/// 处理微信通知

/// </summary>

/// <param name="timestamp"></param>

/// <param name="nonce"></param>

/// <param name="signature"></param>

/// <param name="serialNumber"></param>

/// <param name="body"></param>

/// <returns>true 为处理成功,false 为处理失败</returns>

[RemoteService(IsEnabled = false)]

public async Task<bool> Notify(

    string timestamp,

    string nonce,

    string signature,

    string serialNumber,

    string body)

{

    Logger.LogInformation($"接收到微信支付推送的数据");

    var wxClient = WxPayHelper.GetWechatTenpayClient();

    // 验证签名

    if (!wxClient.VerifyEventSignature(timestamp, nonce, body, signature, serialNumber, out Exception error))

    {

        Logger.LogInformation($"签名验证失败，timestamp: {timestamp}, nonce: {nonce}, signature: {signature}, serialNumber: {serialNumber}");

        return false;

    }

    var callbackModel = wxClient.DeserializeEvent(body);

    // 支付成功

    if ("TRANSACTION.SUCCESS".Equals(callbackModel.EventType))

    {

        // 解密数据

        var resource = wxClient.DecryptEventResource<TransactionResource>(callbackModel);

        Logger.LogInformation($"微信支付成功，订单号{resource.OutTradeNumber}");

        // 在这里处理业务逻辑

        return true;

    }

    Logger.LogInformation($"有未处理的微信支付通知，EventType: {callbackModel.EventType}，ID: {callbackModel.Id}");

    //// 退款成功

    //if ("REFUND.SUCCESS".Equals(callbackModel.EventType))

    //{

    //    var resource = wxClient.DecryptEventResource<RefundResource>(callbackModel);

    //}

    return false;

}

**定时任务处理未收到通知的订单**

有时网络原因我们未收微信的通知，这时候就需要主动调用微信来查询订单状态

public class WxPayHandler: ITransientDependency

{

    /// <summary>

    /// 描述: 如果订单过了有效时间还未支付, 则查询订单状态, 如果已支付则更新状态, 如果为支付则关闭订单

    /// </summary>

    /// <returns></returns>

    public async Task Handle() {

        // 获取微信客户端

        var wxClient = WxPayHelper.GetWechatTenpayClient();

        // 获取已经超时还未支付的订单

        var order = await PayOrderRepository.FindAsync(orderId);

        try

        {

            // 查询订单

            var respone = await wxClient.ExecuteGetPayTransactionByOutTradeNumberAsync(new SKIT.FlurlHttpClient.Wechat.TenpayV3.Models.GetPayTransactionByOutTradeNumberRequest()

            {

                OutTradeNumber = order.OrderNumber

            });

            // 如果交易为成功

            if (respone.TradeState == "SUCCESS")

            {

                // 处理业务逻辑

            }

            else

            {

                // 关闭订单

                await CloseOrder(order);

            }

            await uow.CompleteAsync();

        }

        catch {

        }

    }

    /// <summary>

    /// 关闭订单

    /// </summary>

    /// <param name="order"></param>

    /// <returns></returns>

    protected async Task CloseOrder(PayOrder order) {

        // 关闭微信订单

        var wxClient = WxPayHelper.GetWechatTenpayClient();

        var respone = await wxClient.ExecuteClosePayTransactionAsync(new SKIT.FlurlHttpClient.Wechat.TenpayV3.Models.ClosePayTransactionRequest()

        {

            OutTradeNumber = order.OrderNumber

        });

        // 成功关闭订单

        if (respone.IsSuccessful())

        {

            Logger.LogInformation($"微信关闭订单成功,订单号:{order.OrderNumber}");

        }

        // 当前订单已关闭

        if (respone.ErrorCode == "ORDER\_CLOSED")

        {

            Logger.LogInformation($"微信关闭订单成功,订单号:{order.OrderNumber}");

        }

        // 订单不存在

        if (respone.ErrorCode == "ORDERNOTEXIST")

        {

            Logger.LogInformation($"微信关闭订单成功,订单号:{order.OrderNumber}");

        }

    }

}