

闪聚支付 第2章 讲义-支付参数配置

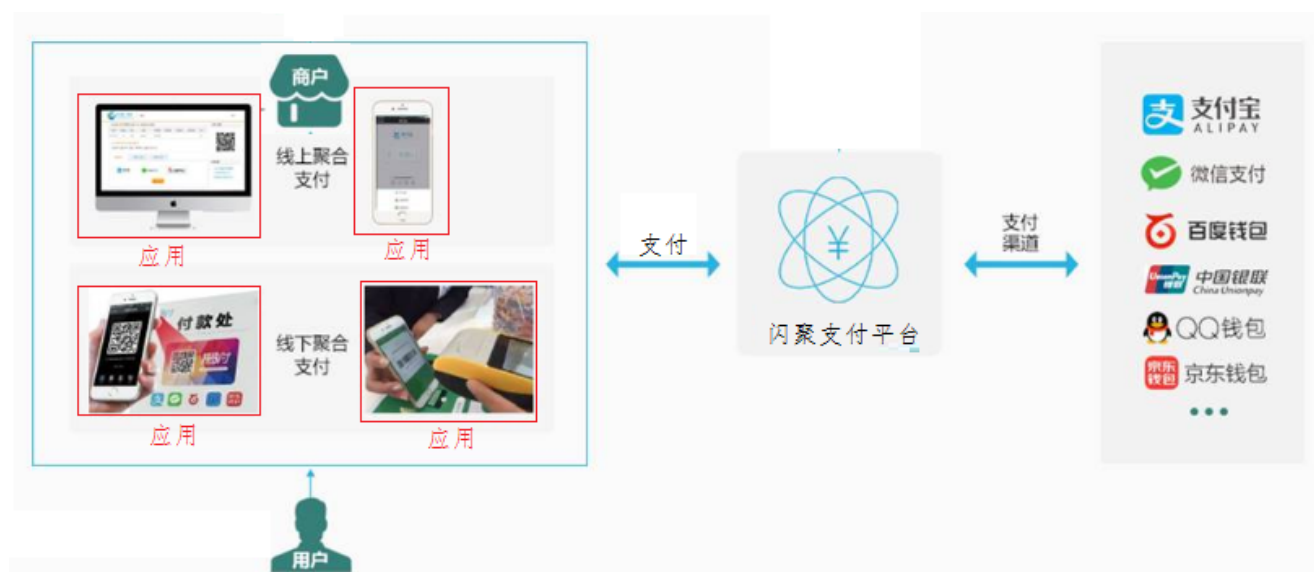
1 需求概述

1.1 基础概念

1.1.1 理解应用

商户资质审核通过后就可以使用闪聚支付平台提供的服务，闪聚支付平台所提供的基础服务正是聚合支付。聚合支付就是将微信、支付宝等支付渠道汇聚为一个支付通道供商户使用，如下图：

闪聚支付平台提供线上支付和线下支付两种方式，线上支付可通过手机和PC完成，线下支付可通过扫码完成。



- 1、闪聚支付平台对接微信、支付宝等众多支付渠道。
- 2、商户创建自己的应用
- 3、用户在使用商户某个应用时发起支付到闪聚支付平台
- 4、闪聚支付平台根据用户的支付请求使用具体的支付渠道完成支付。

支付渠道是什么？

是指微信、支付宝等第三方支付机构提供的支付渠道。闪聚支付平台是要聚合这些支付渠道，为用户提供一个支付通道。

应用是什么？

应用是商户在闪聚支付平台创建的业务标识，比如：商户在自己的XX电商网站使用闪聚支付则会创建“XX电商网站应用”，商户在自己经营的餐厅使用闪聚支付则会创建“XX餐厅应用”。

应用有什么用？

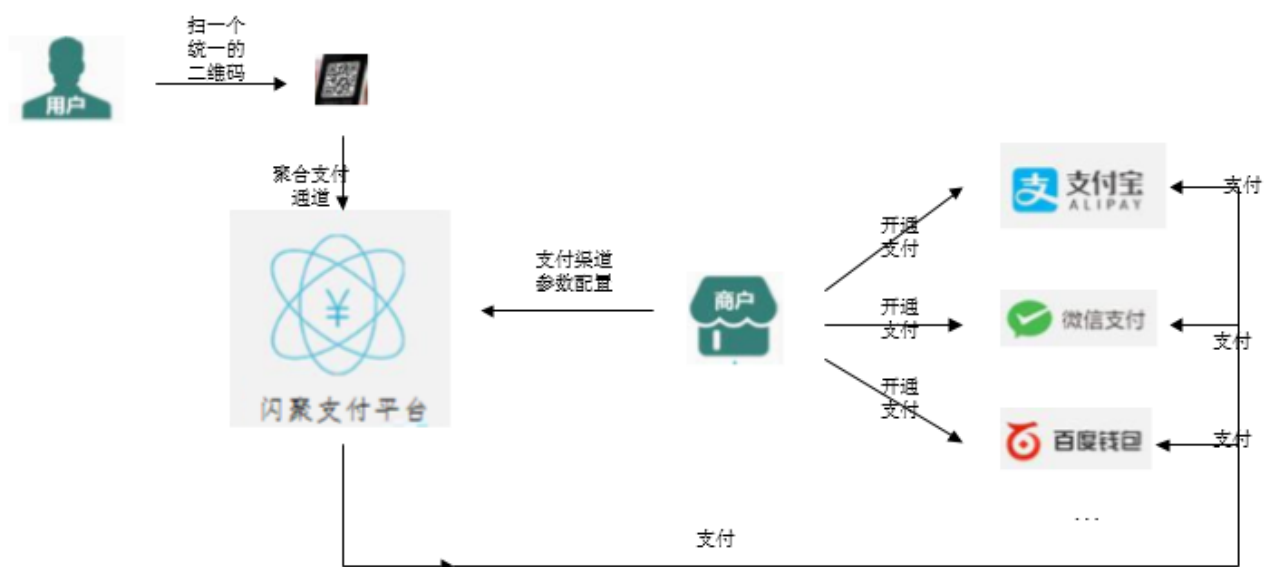
用户是基于某个应用完成的支付，用户在商户的餐厅支付则支付订单会隶属于“XX餐厅应用”下，用户在XX电商网站支付则支付订单会隶属于“XX电商网站应用”下。

闪聚支付平台通过应用来管理商户的支付订单，实现按业务进行订单管理、财务数据统计等功能。比如：可以统计“XX餐厅应用”下的支付订单，统计“XX电商网站应用”下的订单信息。

1.1.2 理解支付渠道参数配置

闪聚支付平台作为一个中介，为了给用户提供更便利的支付体验而聚合了微信、支付宝等第三方支付渠道为一个支付通道，用户通过闪聚支付平台完成支付，闪聚支付平台最终会请求第三方支付渠道完成支付。

所以，商户不仅是闪聚平台的商户，还是第三方支付机构的商户，商户要使用闪聚支付平台就需要开通微信、支付宝等支付渠道，然后在闪聚支付平台配置支付渠道参数，如下图：



一、整体流程如下：

1、商户在支付宝、微信开通支付

下图是商户在支付宝开通支付后的配置参数，包括Appid、密钥等。（在支付章节详细介绍这些参数）

APPID ¹	20161010000000000000000000000000
支付宝网关 ¹	https://openapi.alipaydev.com/gateway.do
RSA2(SHA256)密钥(推荐) ¹	设置/查看
RSA(SHA1)密钥 ¹	查看应用公钥 查看支付宝公钥

2、商户在闪聚支付平台配置支付渠道参数

商户把支付宝、微信等支付渠道的参数配置在闪聚支付平台。

第一步在第三方支付渠道开通支付，商户将第三方支付渠道的APPID、密钥等信息配置在闪聚支付平台。

3、闪聚支付平台为商户生成一个支付二维码

如果不使用闪聚支付平台商户要分别在支付宝、微信生成不同的二维码

4、用户扫二维码完成支付

当用户用支付宝扫描二维码则自动用支付宝完成支付，当使用微信扫描二维码则自动打开微信进行支付。

二、商户应该配置哪些第三方支付渠道的参数呢？

1、首先理解服务类型

服务类型是闪聚支付平台为商户提供的聚合支付服务通道，共分为线上和线下两大类：

线上支付服务通道：

1) 手机APP支付

2) PC网页支付

3) 手机网页支付

4) 小程序支付

等

线下支付服务通道：

1) 收款码支付(C扫B)

商户出示收款码，用户扫收款码完成支付。



2) B扫C，顾客出示付款码，商户扫描付款码



2、商户为应用绑定服务类型

前边学习了“应用”的概念，用户是基于某个应用进行支付，商户为应用绑定服务类型。比如：商户为“XX餐厅应用”绑定服务类型为收款码支付，则用户可以C扫B支付；商户为“XX电商网站应用”指定服务类型为“手机网页支付”，则用户可以通过 手机端在 XX电商网站完成支付。一个应用可以指定多个服务类型。

C扫B支付：顾客扫商户出示的二维码完成支付。

3、配置支付渠道参数

第三方支付渠道提供多种支付方式，比如：微信提供如下支付方式和支付宝提供的支付方式。

微信支付方式：

 <p>付款码支付</p> <p>用户打开微信钱包-付款码的界面，商户扫码后提交完成支付</p>	 <p>JSAPI支付</p> <p>用户通过微信扫码、关注公众号等方式进入商家H5页面，并在微信内调用JSSDK完成支付</p>	 <p>Native支付</p> <p>用户打开“微信扫一扫”，扫描商户的二维码后完成支付</p>	 <p>APP支付</p> <p>商户APP中集成微信SDK，用户点击后跳转到微信内完成支付</p>
 <p>H5支付</p> <p>用户在微信以外的手机浏览器请求微信支付的场景唤起微信支付</p>	 <p>小程序支付</p> <p>用户在微信小程序中使用微信支付的场景</p>	 <p>人脸支付</p> <p>无需掏出手机，刷脸完成支付，适合线下各种场景</p>	

支付宝支付方式：

 当面付 商户扫码收款，消费者扫码付款	 APP支付 接入支付宝SDK，用户支付时唤起支付宝完成支付	 手机网站支付 移动端唤起支付宝钱包或网页收银台完成收银	 电脑网站支付 用户通过支付宝完成支付，交易款项即时到账
 芝麻信用 查询用户信用评分是否高于目标分值	 商户会员卡 商户会员营销基础能力，满足商户个性化营销需求	 安全检测 有效识别问题客户，帮助商户降低业务风险	 全部文档 查看更多文档

商户需要根据应用所绑定的服务类型来配置支付渠道的参数，比如：应用绑定的服务类型是“收款码支付（c扫b）”则需要配置微信的“JSAPI支付”和支付宝的“手机网站支付”参数，如果闪聚支付还聚合了百度、京东等第三方支付渠道，且商户还希望顾客可以用百度、京东的App完成支付，此时商户就需要配置百度、京东所提供的“手机网页支付”参数。

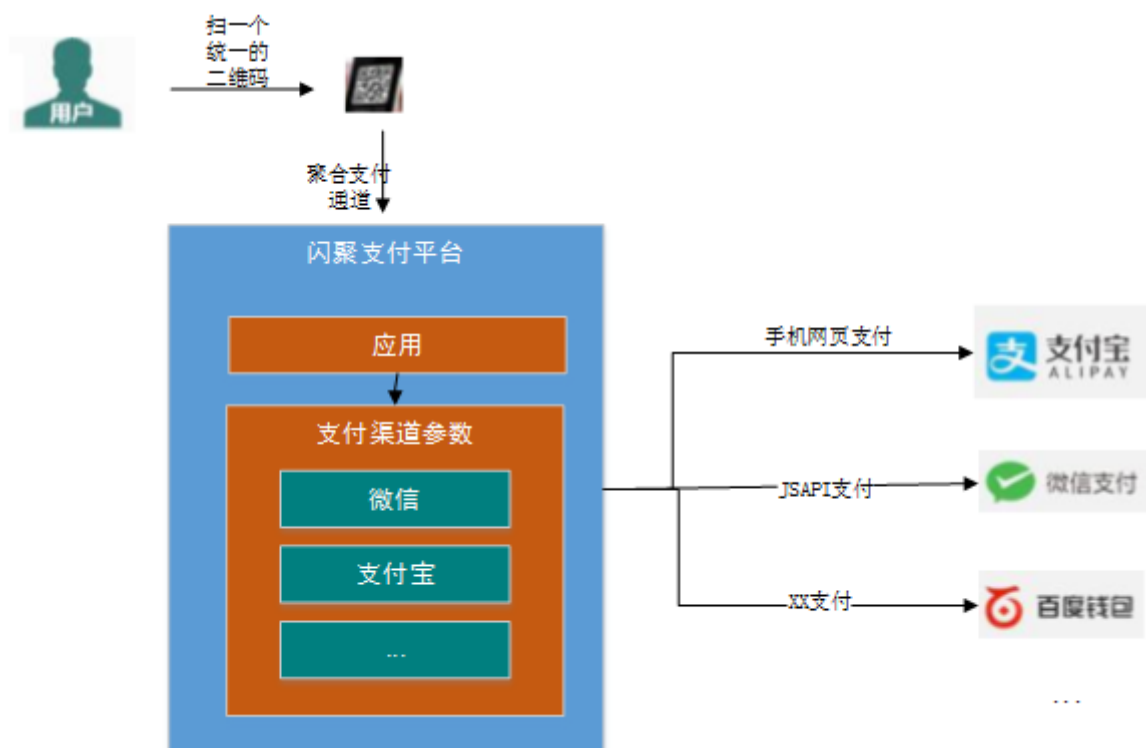
微信JSAPI支付：微信提供的内嵌于微信App内的网页支付，可用于微信公众号支付。

支付宝手机网站支付：支付宝提供的用于手机网页支付方式。

三、总结

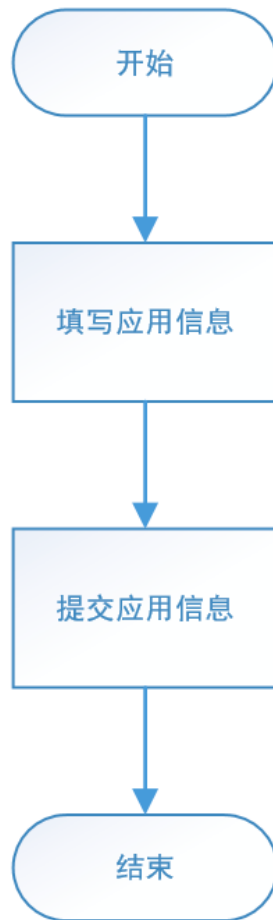
下图展示了闪聚支付平台下应用及支付渠道参数的关系：

- 1、应用是商户创建的业务标识，顾客的每次支付都隶属于某个应用。
- 2、应用绑定闪聚支付平台提供的服务类型。
- 3、根据所绑定的服务类型，需要为应用配置支付渠道参数。



1.2 创建应用

创建应用的流程如下：



1. 填写应用基本信息

应用设置

基础信息	服务类型	配置参数
------	------	------

应用名称

应用公钥

保存

2. 点击保存信息

1.3 支付渠道参数配置

下图是支付渠道参数的配置流程：



1) 应用创建成功后，会自动跳转到绑定服务类型页面

The screenshot shows the 'Payment Application Management' interface. At the top, there's a navigation bar with '闪聚支付' (Shanju Payment) and '账户管理' (Account Management). Below it, the breadcrumb is '闪聚支付 / 支付应用管理'. The main content area is titled '应用设置' (Application Settings) and has three tabs: '基础信息' (Basic Information), '服务类型' (Service Type), and '配置参数' (Configuration Parameters). The '服务类型' tab is selected. It displays a table with two rows of payment applications.

支付ID	支付名称	支付编码	操作
1	闪聚B扫C	shanju_b2c	开启服务 配置实际支付渠道
2	闪聚C扫B	shanju_c2b	开启服务 配置实际支付渠道

2) 点击开启服务为应用绑定服务类型

闪聚支付 账户管理 支付应用管理 组织管理

闪聚支付 / 支付应用管理 亲，欢迎您！ | 退出

应用设置

☒ 基础信息 > ☐ 服务类型 > ☐ 配置参数

支付ID	支付名称	支付编码	操作
1	闪聚B扫C	shanju_b2c	开启服务 配置实际支付渠道
2	闪聚C扫B	shanju_c2b	开启服务 配置实际支付渠道

3) 开启服务后，点击配置实际支付渠道按钮进入参数配置页面

闪聚支付 账户管理 支付应用管理 组织管理

闪聚支付 / 支付应用管理 亲，欢迎您！ | 退出

应用设置

☒ 基础信息 > ☐ 服务类型 > ☐ 配置参数

支付ID	支付名称	支付编码	操作
1	闪聚B扫C	shanju_b2c	开启服务 配置实际支付渠道
2	闪聚C扫B	shanju_c2b	开启服务 配置实际支付渠道

4) 配置参数页面会显示对应服务类型下的原始支付渠道

闪聚支付 账户管理 支付应用管理 组织管理

闪聚支付 / 支付应用管理 亲，欢迎您！ | 退出

应用设置

☒ 基础信息 > ☒ 服务类型 > ☐ 配置参数

支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

5) 点击配置参数按钮，为指定原始支付渠道配置

闪聚支付

账户管理

支付应用管理

组织管理

闪聚支付 / 支付应用管理

亲, 欢迎您! | 退出

应用设置

基础信息

服务类型

配置参数

支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

配置名称:

请输入配置名称

调用配

appId:

请输入appId

应用秘钥:

请输入应用秘钥

安全码:

请输入安全码

同步通知地址:

请输入同步通知地址

合作者PID:

请输入合作者PID

支付秘钥:

请输入支付秘钥

保存

6) 填写支付宝或微信的支付参数

应用设置

基础信息

服务类型

配置参数

支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

配置名称:

C扫B-微信

调用配

appId:

wxd2bf2dba2e86a8c7

应用秘钥:

cec1a9185ad435abe1bcd4b93f7ef2e

安全码:

95fe355daca50f1ae82f0865c2ce87c8

同步通知地址:

http://127.0.0.1:56010/payment-receiver/

合作者PID:

1502570431

支付秘钥:

95fe355daca50f1ae82f0865c2ce87c8

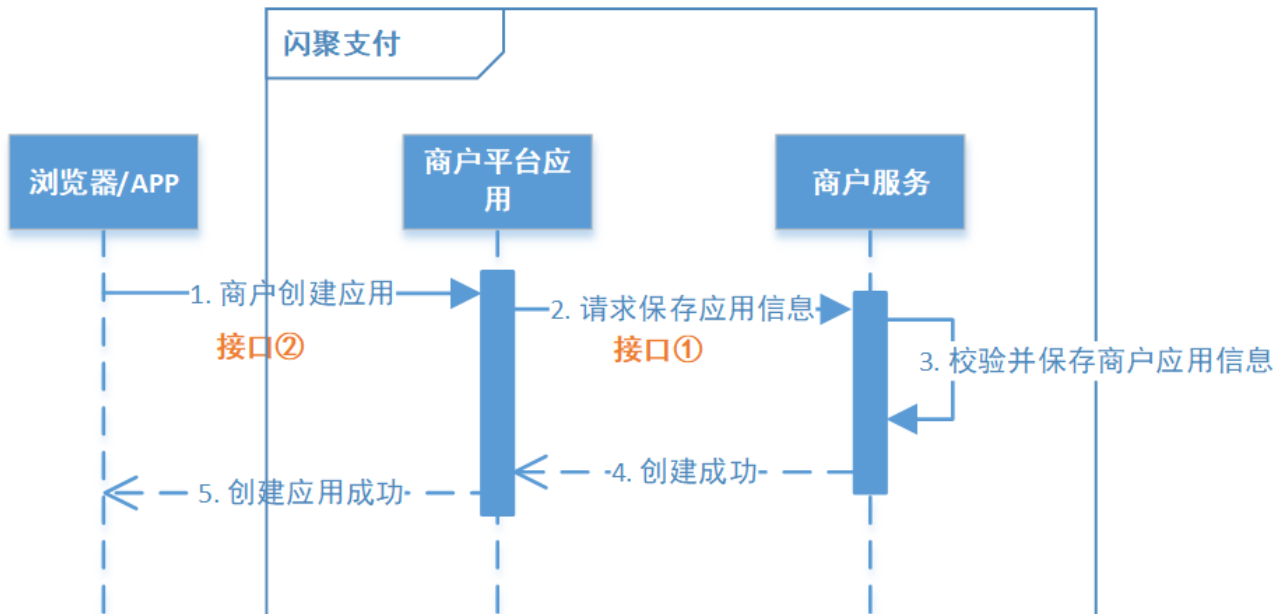
保存

4 商户应用创建

4.1 需求分析

4.1.1 系统交互流程

商户应用创建交互流程如下：



1. 前端携带应用信息请求商户平台应用
2. 请求商户服务保存应用信息
3. 商户服务校验并保存商户应用
4. 返回前端创建成功

4.2 应用创建

4.2.1 商户服务创建应用接口(接口①)

4.2.1.1 接口定义

1、接口描述

- 1) 校验商户是否通过资质审核

如果商户资质审核没有通过不允许创建应用。

- 2) 生成应用ID

应用Id使用UUID方式生成。

- 3) 保存商户应用信息

应用名称需要校验唯一性。

2、接口定义如下：

应用信息保存至商户数据库的app表，根据表字段定义DTO属性。

定义AppDTO：

```
@ApiModel(value = "AppDTO", description = "应用信息")
@Data
public class AppDTO implements Serializable {

    @ApiModelProperty("应用id，新增时无需传入")
    private String appId;

    @ApiModelProperty("应用名称")
    private String appName;

    @ApiModelProperty("商户id")
    private Long merchantId;

    @ApiModelProperty("应用公钥")
    private String publicKey;

    @ApiModelProperty("支付回调应用的url，创建时可不填")
    private String notifyUrl;

}
```

在AppService下定义createApp接口：

```
/**
 * 商户下创建应用
 * @return
 */
AppDTO createApp(Long merchantId, AppDTO app) throws BusinessException;
```

4.2.1.2 接口实现

定义AppCovert负责对象转换

```
@Mapper
public interface AppCovert {

    AppCovert INSTANCE = Mappers.getMapper(AppCovert.class);

    AppDTO entity2dto(App entity);

    App dto2entity(AppDTO dto);

    List<AppDTO> listentity2dto(List<App> app);

}
```

在AppServiceImpl实现createApp接口实现：



```
package com.shanjupay.merchant.service;

@org.apache.dubbo.config.annotation.Service
public class AppServiceImpl implements AppService {

    @Autowired
    private AppMapper appMapper;

    @Autowired
    private MerchantMapper merchantMapper;

    @Override
    public AppDTO createApp(Long merchantId, AppDTO app) {
        //校验商户是否通过资质审核
        Merchant merchant = merchantMapper.selectById(merchantId);
        if (merchant == null) {
            throw new BusinessException(CommonErrorCode.E_200002);
        }
        if (!"2".equals(merchant.getAuditStatus())) {
            throw new BusinessException(CommonErrorCode.E_200003);
        }

        if(isExistAppName(app.getAppName())){
            throw new BusinessException(CommonErrorCode.E_200004);
        }

        //保存应用信息
        app.setAppId(RandomUuidUtil.getUUID());
        app.setMerchantId(merchant.getId());
        App entity = AppCovert.INSTANCE.dto2entity(app);
        appMapper.insert(entity);
        return AppCovert.INSTANCE.entity2dto(entity);
    }

    /**
     * 校验应用名是否已被使用
     * @param appName
     * @return
     */
    public Boolean isExistAppName(String appName) {
        Integer count = appMapper.selectCount(new QueryWrapper<App>
        ().lambda().eq(App::getAppName, appName));
        return count.intValue() > 0;
    }
}
```

4.2.2 商户平台应用创建应用接口(接口②)

4.2.2.1 接口定义

1、接口描述

1) 获取当前登录商户ID

2) 请求商户服务保存应用信息

2、接口定义如下：

定义AppController类，并且定义createApp方法：

```
package com.shanjupay.merchant.controller;

@Api(value = "商户平台-应用管理", tags = "商户平台-应用相关", description = "商户平台-应用相关")
@RestController
public class AppController {

    @Reference
    private AppService appService;

    @ApiOperation("商户创建应用")
    @ApiImplicitParams({
        @ApiImplicitParam(name = "app", value = "应用信息", required = true, dataType = "AppDTO", paramType = "body")})
    @PostMapping(value = "/my/apps")
    public AppDTO createApp(@RequestBody AppDTO app) {
        Long merchantId = SecurityUtil.getMerchantId();
        return appService.createApp(merchantId, app);
    }
}
```

4.2.2.2 接口测试

1、准备token

用之前的TokenTemp.java来生成token，注意所指定的商户id必须审核通过。

Key	Value	Description
<input checked="" type="checkbox"/> authorization	Bearer eyJtZXJjaGFudElkIjoxMjA5ODI2Njc4NjM1Nzk4NT...	

2、添加应用

创建应用

POST	http://localhost:57010/merchant/my/apps	Params	Send
Authorization Headers (2) Body Pre-request Script Tests			
form-data x-www-form-urlencoded raw binary JSON (application/json)			
<pre>1 { 2 "appName": "这里输出应用名称", 3 "notifyUrl": "string", 4 "publicKey": "string" 5 }</pre>			

4.3 应用查询

4.3.1 商户服务应用查询接口

4.3.1.1 接口定义

1、根据商户ID查询应用

接口定义如下：AppService

```
/**
 * 查询商户下的应用列表
 * @param merchantId
 * @return
 */
List<AppDTO> queryAppByMerchant(Long merchantId) throws BusinessException;
```

2、根据应用ID查询详细信息

接口定义如下：AppService

```
/**
 * 根据业务id查询应用
 * @param id
 * @return
 */
AppDTO getAppById(String id) throws BusinessException;
```

4.3.1.2 接口实现

1、AppServiceImpl

```
@Override
public List<AppDTO> queryAppByMerchant(Long merchantId) {
    List<App> apps = appMapper.selectList(new QueryWrapper<App>
    ().lambda().eq(App::getMerchantId,merchantId));
    List<AppDTO> appDTOS = AppCovert.INSTANCE.listentity2dto(apps);
    return appDTOS;
}

@Override
public AppDTO getAppById(String id) {
    App app = appMapper.selectOne(new QueryWrapper<App>().lambda().eq(App::getAppId, id));
    return AppCovert.INSTANCE.entity2dto(app);
}
```

4.3.2 商户平台应用查询接口

4.3.2.1 接口定义

1、请求商户服务查询商户下的所有应用

接口定义如下：MerchantController

```
@ApiOperation("查询商户下的应用列表")
@GetMapping(value = "/my/apps")
public List<AppDTO> queryMyApps() {
    Long merchantId = SecurityUtil.getMerchantId();
    return appService.queryAppByMerchant(merchantId);
}
```

2、请求商户服务查询应用详细信息

接口定义如下：AppController

```
@ApiOperation("根据appid获取应用的详细信息")
@ApiImplicitParams({
    @ApiImplicitParam(name = "appId", value = "商户应用id", required = true, dataType = "String", paramType = "path")})
@GetMapping(value = "/my/apps/{appId}")
public AppDTO getApp(@PathVariable String appId) {
    return appService.getAppById(appId);
}
```

4.3.2.2 接口测试

1、获取商户的应用列表

获取商户的应用列表

GET	http://localhost:57010/merchant/my/apps	Params	Send
Authorization Headers (1) Body Pre-request Script Tests			
<input checked="" type="checkbox"/>	Key	Value	Description
<input checked="" type="checkbox"/>	authorization	Bearer eyJtZXJjaGFudElkIjoxMTk3NTQyNzQwODg...	
<input type="checkbox"/>	New key	Value	Description
Body Cookies (4) Headers (3) Test Results Status: 200 OK Time: 5			
Pretty Raw Preview JSON			

2、根据appId获取应用的详细信息

▶ 根据appid获取应用的详细信息

GET ▾	http://localhost:57010/merchant/my/apps/586bef7090bd4f339928b5e4bfe500cb	Params	Send ▾
-------	--	--------	--------

Authorization Headers (1) Body Pre-request Script Tests

TYPE

Inherit auth from parent ▾

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

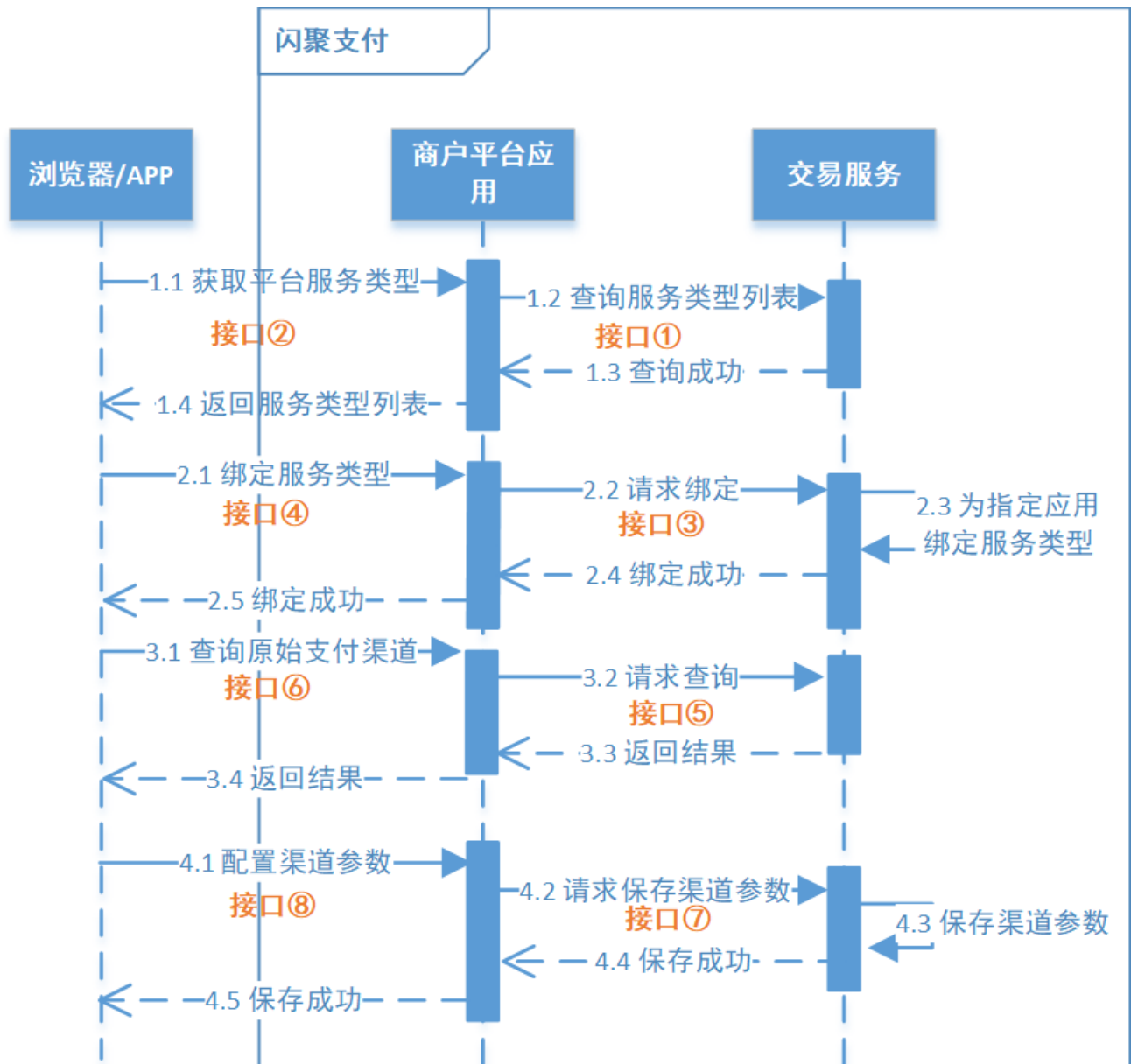
This request is using an authorization helper from collection [闪鑿支付-m1](#).

5 支付渠道参数配置

5.1 需求分析

5.1.1 系统交互流程

商户渠道参数配置交互流程如下：



交易服务职责：提供支付渠道参数配置、订单、发起支付、转账、退款等功能

交互流程如下：

第一阶段：应用绑定服务类型

1. 前端请求商户平台应用获取平台支持的所有服务类型列表
2. 请求交易服务查询列表
3. 返回服务类型列表给前端



4. 前端选择要绑定的服务类型请求商户平台应用

5. 请求交易服务绑定服务类型



闪聚支付 / 支付应用管理

应用设置

基础信息 > 服务类型 > 配置参数

支付ID	支付名称	支付编码	操作
1	闪聚B扫C	shanju_b2c	开启服务 配置实际支付渠道
2	闪聚C扫B	shanju_c2b	开启服务 配置实际支付渠道

6. 返回前端绑定成功

第二阶段：支付渠道参数配置

7. 前端请求获取第三方支付渠道列表



闪聚支付 / 支付应用管理

应用设置

基础信息 > 服务类型 > 配置参数

支付ID	支付名称	支付编码	操作
1	闪聚B扫C	shanju_b2c	开启服务 配置实际支付渠道
2	闪聚C扫B	shanju_c2b	开启服务 配置实际支付渠道

8. 请求交易服务获取列表

9. 返回结果给前端



闪聚支付 / 支付应用管理

应用设置

基础信息 > 服务类型 > 配置参数

支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

10. 前端请求配置支付渠道参数

闪聚支付 / 支付应用管理 账户管理 支付应用管理 组织管理

亲，欢迎您！ | 退出

应用设置

基础信息 > 服务类型 > 配置参数

支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

配置名称: 调用配1

appid:

应用秘钥:

安全码:

同步通知地址:

合作者PID:

支付秘钥:

保存

11. 商户平台应用请求交易服务保存参数配置

应用设置

基础信息 > 服务类型 > 配置参数

支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

配置名称: 调用配1

appid:

应用秘钥:

安全码:

同步通知地址:

合作者PID:

支付秘钥:

保存

12. 返回前端保存成功

5.1.2 基础数据

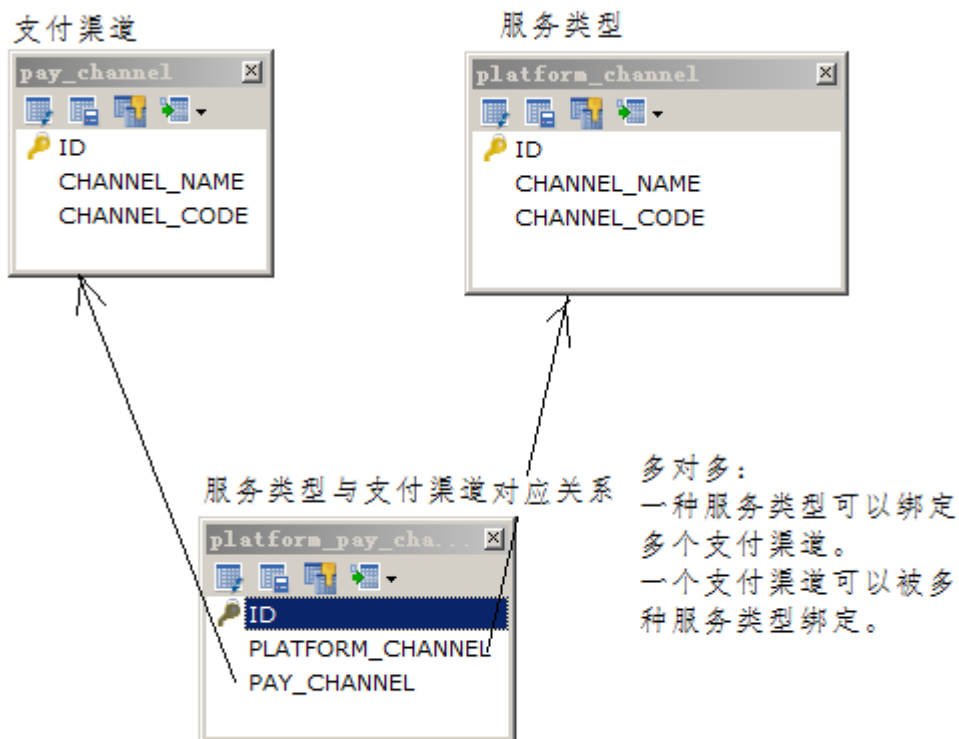
初始化平台基础数据：

platform_channel：平台服务类型

pay_channel：第三方支付渠道

platform_pay_channel：平台服务类型对应第三方支付渠道

平台服务类型应根据自身特点对接第三方支付渠道，例如：C扫B服务类型则需要对接微信SAPI接口和支付宝手机网站支付接口。



```
use shanjupay_transaction;

LOCK TABLES `platform_channel` WRITE;
/*!40000 ALTER TABLE `platform_channel` DISABLE KEYS */;

INSERT INTO `platform_channel` (`ID`, `CHANNEL_NAME`, `CHANNEL_CODE`)
VALUES
  (1, '闪聚B扫C', 'shanju_b2c'),
  (2, '闪聚C扫B', 'shanju_c2b'),
  (3, '微信Native支付', 'wx_native'),
  (4, '支付宝手机网站支付', 'alipay_wap');

/*!40000 ALTER TABLE `platform_channel` ENABLE KEYS */;
UNLOCK TABLES;

LOCK TABLES `pay_channel` WRITE;
/*!40000 ALTER TABLE `pay_channel` DISABLE KEYS */;

INSERT INTO `pay_channel` (`ID`, `CHANNEL_NAME`, `CHANNEL_CODE`)
VALUES
  (1, '微信JSAPI', 'WX_JSAPI'),
  (2, '支付宝手机网站支付', 'ALIPAY_WAP'),
  (3, '支付宝条码支付', 'ALIPAY_BAR_CODE'),
  (4, '微信付款码支付', 'WX_MICROPAY');
```

```
(5,'微信native支付','WX_NATIVE');

/*!40000 ALTER TABLE `pay_channel` ENABLE KEYS */;
UNLOCK TABLES;

LOCK TABLES `platform_pay_channel` WRITE;
/*!40000 ALTER TABLE `platform_pay_channel` DISABLE KEYS */;

INSERT INTO `platform_pay_channel` (`ID`,`PLATFORM_CHANNEL`,`PAY_CHANNEL`)
VALUES
  (1,'shanju_b2c','WX_MICROPAY'),
  (2,'shanju_b2c','ALIPAY_BAR_CODE'),
  (3,'wx_native','WX_NATIVE'),
  (4,'alipay_wap','ALIPAY_WAP'),
  (5,'shanju_c2b','WX_JSAPI'),
  (6,'shanju_c2b','ALIPAY_WAP');

/*!40000 ALTER TABLE `platform_pay_channel` ENABLE KEYS */;
UNLOCK TABLES;
```

5.2 搭建交易服务工程

5.2.1 交易服务介绍

交易服务：提供渠道参数配置、订单、发起支付、转账、退款等功能

工程名	职责
交易服务API(shanjupay-transaction-api)	定义交易服务提供的接口
交易服务(shanjupay-transaction-service)	实现交易服务的接口实现

5.2.2 搭建工程

1. 复制提供的shanjupay-transaction目录到shanjupay根目录，导入shanjupay-transaction及api和service三个工程。
2. 添加Module到IDEA中



3. 在Nacos中添加transaction-service.yaml配置，Group: SHANJUPAY_GROUP

```
# 覆盖spring-boot-http.yaml的项目
server:
  servlet:
    context-path: /transaction

# 覆盖spring-boot-starter-druid.yaml的项目
spring:
  datasource:
    druid:
      url: jdbc:mysql://127.0.0.1:3306/shanjupay_transaction?
      useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai&useSSL=false
      username: root
      password: yourpassword

# 覆盖spring-boot-mybatis-plus.yaml的项目
mybatis-plus:
  typeAliasesPackage: com.shanjupay.transaction.entity
  mapper-locations: classpath:com/shanjupay/*/mapper/*.xml
```

* Data ID: transaction-service.yaml

* Group: SHANJUPAY_GROUP

更多高级选项

描述: 交易中心

Beta发布: ☐ 默认不要勾选。

配置格式: ☐ TEXT ☐ JSON ☐ XML ☒ YAML ☐ HTML ☐ Properties

配置内容:

```
1 # 覆盖spring-boot-http.yaml的项目
2 server:
3   servlet:
4     context-path: /transaction
5
6 # 覆盖spring-boot-starter-druid.yaml的项目
```

4. 打开shanjupay-transaction-service工程的bootstrap.yml，将其中的namespace替换为dev命名空间的ID

```
13 cloud:
14   nacos:
15     discovery:
16       server-addr: ${nacos.server.addr}
17       namespace: a1f8e863-3117-48c4-9dd3-e9ddc2af90a8
18       cluster-name: DEFAULT
19     config:
20       server-addr: ${nacos.server.addr} # 配置中心地址
21       file-extension: yaml
22       namespace: a1f8e863-3117-48c4-9dd3-e9ddc2af90a8 #2ed00aaa-b760-417d
23       group: SHANJUPAY_GROUP # 聚合支付业务组
24       ext-config:
25         -
26         refresh: true
```

5. 启动TransactionBootstrap测试

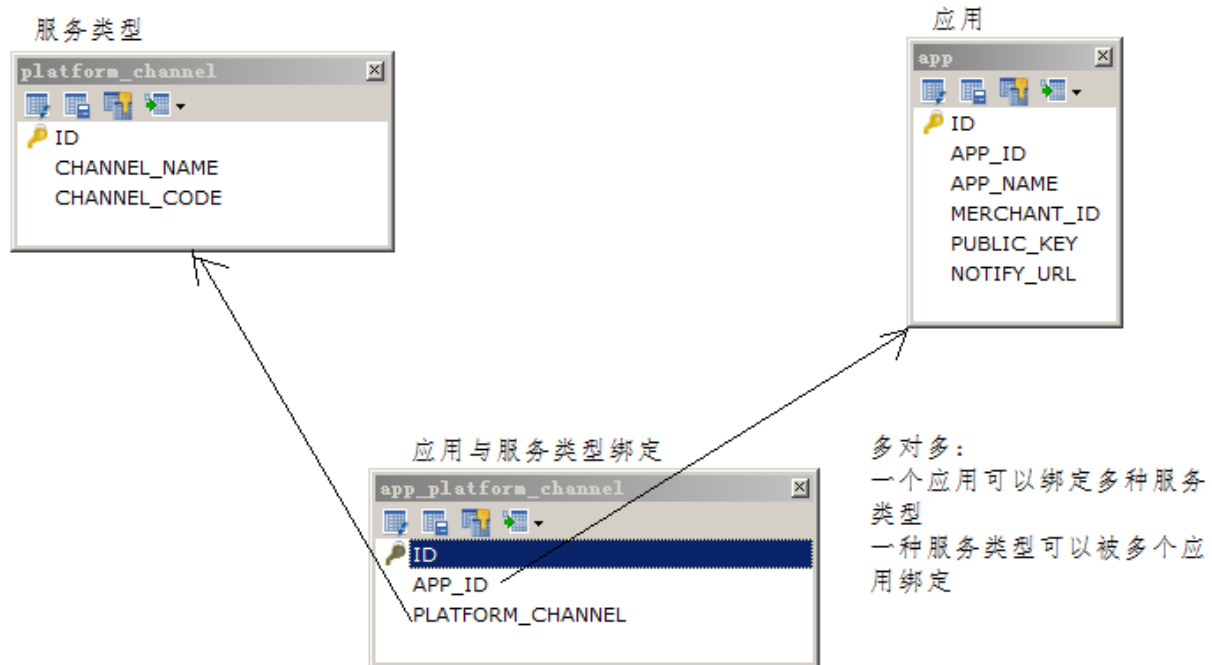
```
bootstrap x TransactionBootstrap x
1 package com.shanjupay.transaction;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6
7 @SpringBootApplication
8 @EnableDiscoveryClient
9 public class TransactionBootstrap {
10     public static void main(String[] args) { SpringApplication.run(TransactionB
13 }
```

启动成功后观察nacos的服务列表，看transaction-service是否注册服务成功，如果注册成功则说明交易服务启动成功。

5.3 应用绑定服务类型

5.3.1 系统设计

为应用绑定服务类型即指定应用使用哪些服务类型，数据模型设计如下：



5.3.2 交易服务获取平台服务类型(接口①)

绑定服务类型页面，页面中列出服务类型。



5.3.1.1 接口定义

- 1、接口描述：查询平台支持的所有服务类型
- 2、接口定义如下：PayChannelService


```
package com.shanjupay.transaction.api;

/**
 * 支付渠道服务 管理平台支付渠道，原始支付渠道，以及相关配置
 */
public interface PayChannelService {

    /**
     * 获取平台服务类型
     * @return
     */
    List<PlatformChannelDTO> queryPlatformChannel() throws BusinessException;
}
```

5.3.1.2 接口实现

定义PayChannelServiceImpl类及queryPlatformChannel实现方法：

```
package com.shanjupay.transaction.service;

import com.shanjupay.transaction.api.PayChannelService;
import com.shanjupay.transaction.api.dto.PlatformChannelDTO;
import com.shanjupay.transaction.convert.PlatformChannelConvert;
import com.shanjupay.transaction.entity.PlatformChannel;
import com.shanjupay.transaction.mapper.PlatformChannelMapper;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.List;

@org.apache.dubbo.config.annotation.Service
public class PayChannelServiceImpl implements PayChannelService {

    @Autowired
    private PlatformChannelMapper platformChannelMapper;

    @Override
    public List<PlatformChannelDTO> queryPlatformChannel() {
        List<PlatformChannel> platformChannels = platformChannelMapper.selectList(null);
        List<PlatformChannelDTO> platformChannelDTOS =
            PlatformChannelConvert.INSTANCE.listentity2listdto(platformChannels);
        return platformChannelDTOS;
    }
}
```

5.3.3 商户平台应用获取平台服务类型(接口②)

5.3.3.1 接口定义

在商户平台应用工程添加依赖：

```
<dependency>
  <groupId>com.shanjupay</groupId>
  <artifactId>shanjupay-transaction-api</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
```

1、接口描述：请求交易服务查询平台支持的所有服务类型

2、接口定义如下：PlatformParamController

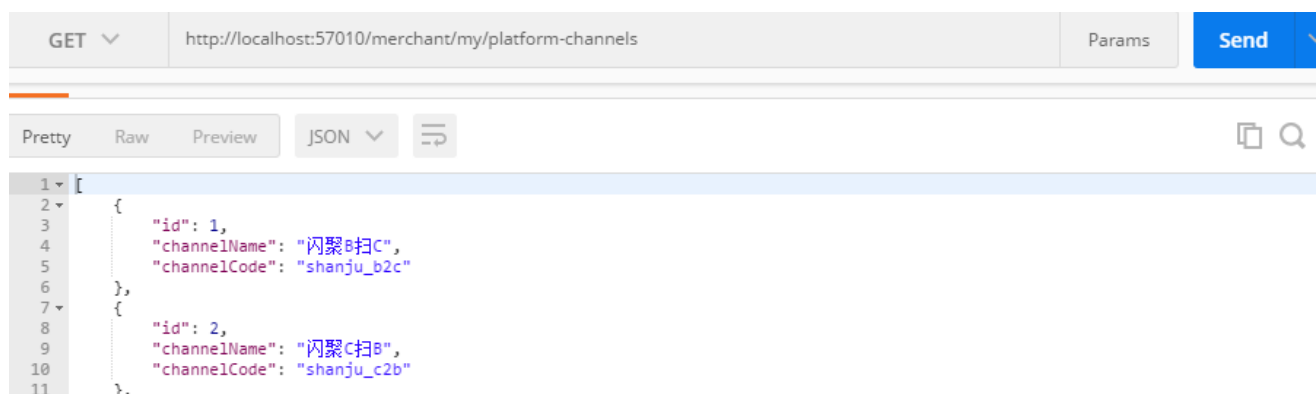
```
package com.shanjupay.merchant.controller;

@Api(value = "商户平台-渠道和支付参数相关", tags = "商户平台-渠道和支付参数", description = "商户平台-渠道和支付参数相关")
@Slf4j
@RestController
public class PlatformParamController {

    @Reference
    private PayChannelService payChannelService;

    @ApiOperation("获取平台服务类型")
    @GetMapping(value="/my/platform-channels")
    public List<PlatformChannelDTO> queryPlatformChannel(){
        return payChannelService.queryPlatformChannel();
    }
}
```

5.3.3.2 接口测试



The image shows a REST client interface with a GET request to `http://localhost:57010/merchant/my/platform-channels`. The response is displayed in JSON format, showing a list of two channel objects. The first object has `id: 1`, `channelName: "闪聚B扫C"`, and `channelCode: "shanju_b2c"`. The second object has `id: 2`, `channelName: "闪聚C扫B"`, and `channelCode: "shanju_c2b"`.

```
[{"id": 1, "channelName": "闪聚B扫C", "channelCode": "shanju_b2c"}, {"id": 2, "channelName": "闪聚C扫B", "channelCode": "shanju_c2b"}]
```

5.3.4 交易服务绑定服务类型接口(接口③)

点击开启服务为应用绑定服务类型



5.3.4.1 接口定义

1、接口描述：

- 1) 查询出指定应用是否已绑定选定的服务类型
- 2) 如果该应用没有绑定该 服务类型则进行绑定

2、接口定义如下：

在PayChannelService接口中定义bindPlatformChannelForApp

```
/**
 * 为app绑定平台服务类型
 * @param appId 应用id
 * @param platformChannelCodes 平台服务类型列表
 */
void bindPlatformChannelForApp(String appId, String platformChannelCodes) throws
BusinessException;
```

5.3.4.2 接口实现

定义PayChannelServiceImpl类及bindPlatformChannelForApp实现方法：

```
package com.shanjupay.transaction.service;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.shanjupay.transaction.api.PayChannelService;
import com.shanjupay.transaction.entity.AppPlatformChannel;
import com.shanjupay.transaction.mapper.AppPlatformChannelMapper;
import org.apache.dubbo.config.annotation.Service;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;

@Service
public class PayChannelServiceImpl implements PayChannelService {

    @Autowired
```

```
private AppPlatformChannelMapper appPlatformChannelMapper;

@Override
@Transactional
public void bindPlatformChannelForApp(String appId, String platformChannelCodes) throws
BusinessException {
    //根据appId和平台服务类型code查询app_platform_channel
    AppPlatformChannel appPlatformChannel = appPlatformChannelMapper.selectOne(new
    LambdaQueryWrapper<AppPlatformChannel>()
        .eq(AppPlatformChannel::getAppId, appId)
        .eq(AppPlatformChannel::getPlatformChannel, platformChannelCodes));
    //如果没有绑定则绑定
    if(appPlatformChannel == null){
        appPlatformChannel = new AppPlatformChannel();
        appPlatformChannel.setAppId(appId);
        appPlatformChannel.setPlatformChannel(platformChannelCodes);
        appPlatformChannelMapper.insert(appPlatformChannel);
    }
}
}
```

5.3.5 商户平台应用绑定服务类型接口(接口④)

5.3.5.1 接口定义

1、接口描述：请求交易服务为指定应用添加服务类型

2、接口定义如下：

在AppController类中定义bindPlatformForApp方法：

```
@Reference
private PayChannelService payChannelService;

@ApiOperation("绑定服务类型")
@PostMapping(value="/my/apps/{appId}/platform-channels")
@ApiImplicitParams({
    @ApiImplicitParam(value = "应用id",name = "appId",dataType = "string",paramType =
"path"),
    @ApiImplicitParam(value = "服务类型code",name = "platformChannelCodes",dataType =
"string",paramType = "query")
})
public void bindPlatformForApp(@PathVariable("appId") String appId,
@RequestParam("platformChannelCodes") String platformChannelCodes){
    payChannelService.bindPlatformChannelForApp(appId,platformChannelCodes);
}
```

5.3.5.2 接口测试

► 为应用绑定服务类型

POST ▼	http://localhost:57010/merchant/my/apps/2a3460850b6a47ebb7b6a7b91f79412c/platform-channels? platformChannelCodes=shanju_c2b	Params	Send ▼
Authorization	Headers	Body	Pre-request Script Tests

TYPE
Inherit auth from parent ▼

5.3.6 交易服务查询服务类型绑定状态

5.3.6.1 接口定义

1、接口描述

1) 查询应用是否已经绑定了某个服务类型

2、接口定义如下

1) PayChannelService

```
/**
 * 应用是否已经绑定了某个服务类型
 * @param appId
 * @param platformChannel
 * @return 已绑定返回1，否则 返回0
 */
int queryAppBindPlatformChannel(String appId,String platformChannel) throws
BusinessException;
```

5.5.5.2 接口实现

在PayChannelServiceImpl中实现queryAppBindPlatformChannel方法：

```
@Override
public int queryAppBindPlatformChannel(String appId, String platformChannel) {
    int count = appPlatformChannelMapper.selectCount(
        new QueryWrapper<AppPlatformChannel>().lambda().eq(AppPlatformChannel::getAppId, appId)
            .eq(AppPlatformChannel::getPlatformChannel, platformChannel));
    //已存在绑定关系返回1
    if (count > 0) {
        return 1;
    } else {
        return 0;
    }
}
```

5.3.7 商户平台查询服务类型绑定状态

5.3.7.1 接口实现

接口描述

查询应用是否已经绑定了某个服务类型

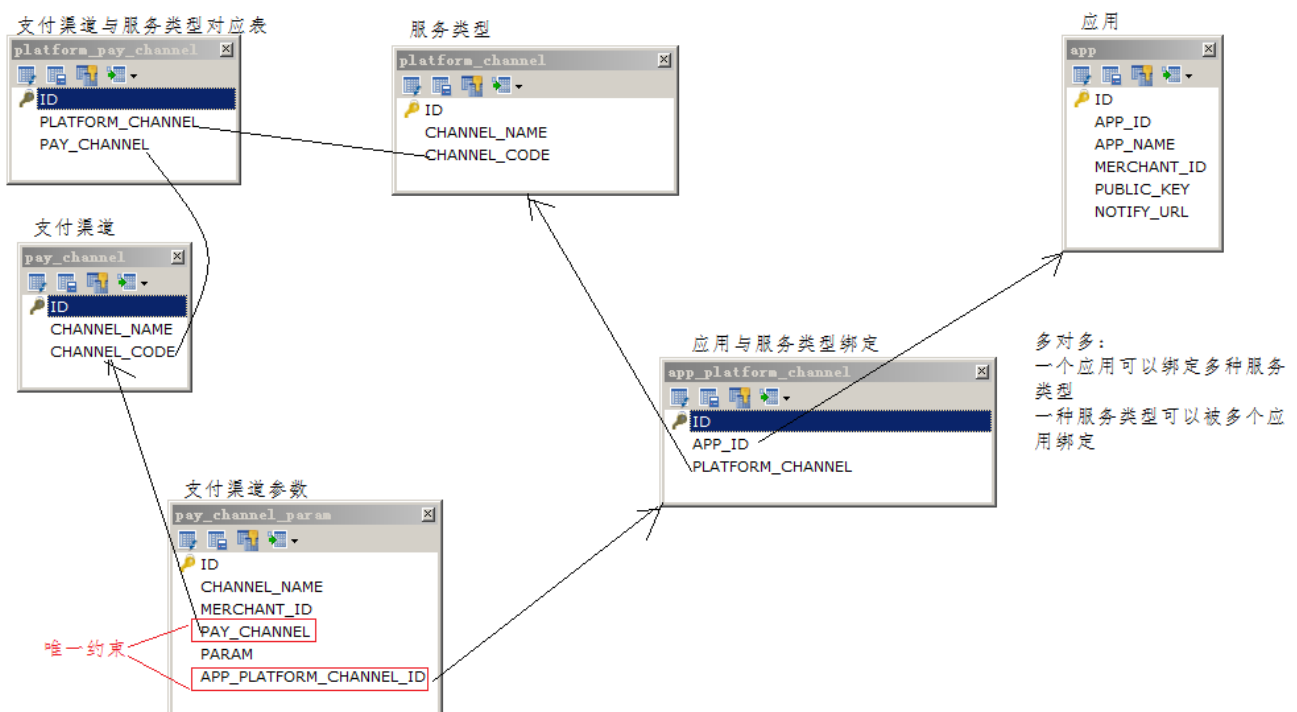
在AppController中定义：

```
@ApiOperation("查询应用是否绑定了某个服务类型")
@ApiImplicitParams({
    @ApiImplicitParam(name = "appId", value = "应用appId", required = true, dataType = "String", paramType = "query"),
    @ApiImplicitParam(name = "platformChannel", value = "服务类型", required = true, dataType = "String", paramType = "query")
})
@GetMapping("/my/merchants/apps/platformchannels")
public int queryAppBindPlatformChannel(@RequestParam String appId, @RequestParam String platformChannel){
    return payChannelService.queryAppBindPlatformChannel(appId,platformChannel);
}
```


5.4 支付渠道参数配置

5.4.1 系统设计

支付渠道参数配置数据模型如下：




支付渠道参数数据存储至支付渠道参数表 (pay_channel_param)

Field	Type	Comment
 ID	bigint(20) NOT NULL	
CHANNEL_NAME	varchar(50) NULL	配置名称
MERCHANT_ID	bigint(20) NULL	商户ID
PAY_CHANNEL	varchar(50) NULL	原始支付渠道编码
PARAM	text NULL	支付参数
APP_PLATFORM_CHANNEL_ID	bigint(20) NULL	应用和服务类型绑定关系id

Indexes (2)

找出多余索引

找到该表的冗余索引。 [了解详情](#)

Indexes	Columns	Index Type
 PRIMARY	ID	Unique
pay_channel_param_index1	PAY_CHANNEL, APP_PLATFORM_CHANNEL_ID	Unique

APP_PLATFORM_CHANNEL_ID：为app_platform_channel表的主键即应用绑定服务类型表的主键，应用加服务类型表示一个APP_PLATFORM_CHANNEL_ID。

APP_PLATFORM_CHANNEL_ID和PAY_CHANNEL唯一约束：即应用、服务类型、第三方支付渠道唯一约束，表示为某应用所绑定的某服务类型的某支付渠道配置参数

例如：

应用app01，服务类型为shanju_c2b，两者在app_platform_channel表示app01应用绑定了shanju_c2b服务类型。

又由于shanju_c2b服务类型对应WX_JSAPI支付渠道，所以在支付渠道参数表pay_channel_param中为应用app01所绑定的服务类型为shanju_c2b配置WX_JSAPI支付渠道参数。

5.4.2 交易服务原始支付渠道查询接口(接口⑤)

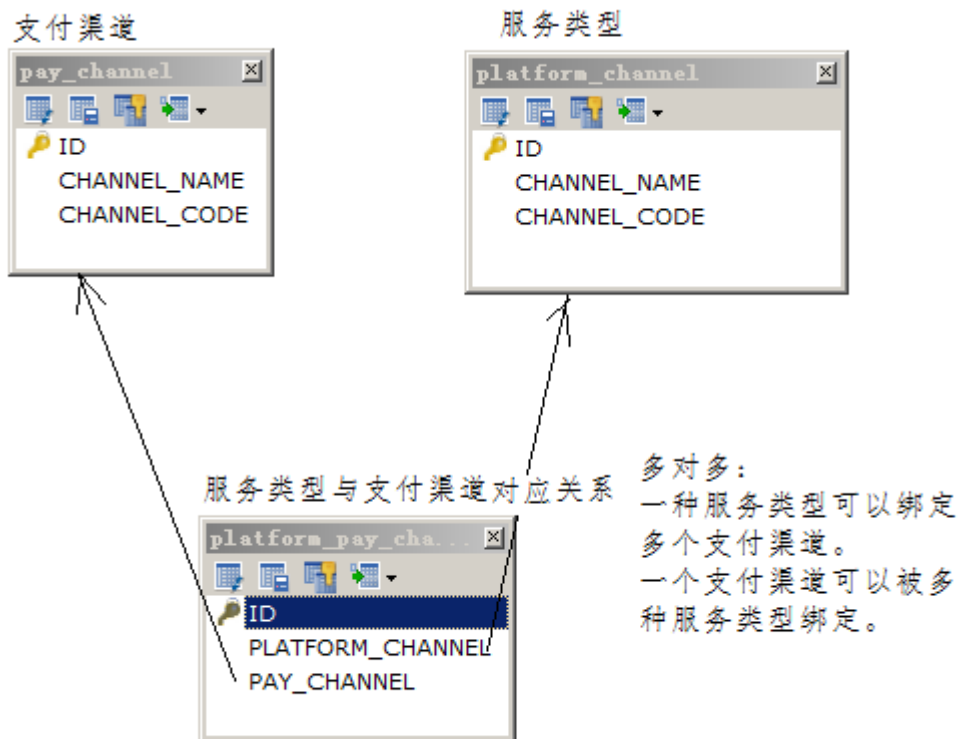
配置参数页面会显示对应服务类型下的原始支付渠道

闪聚支付	账户管理	支付应用管理	组织管理
闪聚支付 / 支付应用管理			
应用设置			
<input checked="" type="radio"/> 基础信息 > <input checked="" type="radio"/> 服务类型 > <input type="radio"/> 配置参数			
支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

5.4.2.1 接口定义

这里是要查询某服务类型下的支付渠道，以便下一步为某支付渠道配置参数。

可从服务类型与支付渠道对应关系表关联查询：



1、接口描述：根据平台服务类型获取支付渠道列表

2、接口定义如下：

在PayChannelService接口中定义queryPayChannelByPlatformChannel：

```
/**
 * 根据平台服务类型获取支付渠道列表
 * @param platformChannelCode
 * @return
 */
List<PayChannelDTO> queryPayChannelByPlatformChannel(String platformChannelCode) throws
BusinessException;
```

5.4.2.2 接口实现

1、在PlatformChannelMapper中定义selectPayChannelByPlatformChannel方法：

```
/**
 * 根据平台服务类型获取原始支付渠道
 * @param platformChannelCode
 * @return
 */
@Select("SELECT " +
        " pay.* " +
        "FROM " +
        " pay_channel pay, " +
```



```
        " platform_pay_channel pac," +
        " platform_channel pla " +
        "WHERE pay.CHANNEL_CODE = pac.PAY_CHANNEL " +
        " AND pla.CHANNEL_CODE = pac.PLATFORM_CHANNEL " +
        " AND pla.CHANNEL_CODE = #{platformChannelCode} ")
    public List<PayChannelDTO> selectPayChannelByPlatformChannel(String platformChannelCode) ;
```

2、在PayChannelServiceImpl类定义queryPayChannelByPlatformChannel实现方法：

```
@Override
public List<PayChannelDTO> queryPayChannelByPlatformChannel(String platformChannelCode) {
    return platformChannelMapper.selectPayChannelByPlatformChannel(platformChannelCode);
}
```

5.4.2.3 接口测试

在交易 服务进行单元 测试，编写单元 测试类

```
package com.shanjupay.transaction;

import com.shanjupay.transaction.api.PayChannelService;
import com.shanjupay.transaction.api.dto.PayChannelDTO;
import lombok.extern.slf4j.Slf4j;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

import java.util.List;

/**
 * @author Administrator
 * @version 1.0
 */
@SpringBootTest
@RunWith(SpringRunner.class)
@Slf4j
public class TestPayChannelService {

    @Autowired
    PayChannelService payChannelService;

    //测试根据服务类型查询支付渠道
    @Test
    public void testqueryPayChannelByPlatformChannel(){
        List<PayChannelDTO> shanju_c2b =
        payChannelService.queryPayChannelByPlatformChannel("shanju_c2b");
        System.out.println(shanju_c2b);
    }
}
```

```
}
```

5.4.3 商户平台应用支付渠道查询接口(接口⑥)

5.4.3.1 接口定义

1、接口描述：根据服务类型查询支付渠道列表

2、接口定义如下：

在PlatformParamController类中定义queryPayChannelByPlatformChannel：

```
@ApiOperation("根据平台服务类型获取支付渠道列表")
@ApiImplicitParams({
    @ApiImplicitParam(name = "platformChannelCode", value = "服务类型编码", required =
true, dataType = "String", paramType = "path")
})
@GetMapping(value="/my/pay-channels/platform-channel/{platformChannelCode}")
public List<PayChannelDTO> queryPayChannelByPlatformChannel(@PathVariable String
platformChannelCode){
    return payChannelService.queryPayChannelByPlatformChannel(platformChannelCode);
}
```

5.4.3.2 接口测试

使用Postman：http://localhost:57010/merchant/my/pay-channels/platform-channel/shanju_c2b

返回值：

```
[
  {
    "id": 1,
    "channelName": "微信JSAPI",
    "channelCode": "WX_JSAPI"
  },
  {
    "id": 2,
    "channelName": "支付宝手机网站支付",
    "channelCode": "ALIPAY_WAP"
  }
]
```

5.4.4 交易服务支付渠道参数配置接口(接口⑦)

为指定原始支付渠道配置

闪聚支付 账户管理 支付应用管理 组织管理

闪聚支付 / 支付应用管理 亲，欢迎您！ | 退出

应用设置

基础信息 服务类型 配置参数

支付ID	支付名称	支付编码	操作
1	微信JSAPI	WX_JSAPI	配置参数
2	支付宝手机网站支付	ALIPAY_WAP	配置参数

配置名称:

请输入配置名称

调用配

appid:

请输入appid

应用密钥:

请输入应用密钥

安全码:

请输入安全码

同步通知地址:

请输入同步通知地址

合作者PID:

请输入合作者PID

支付密钥:

请输入支付密钥

保存

5.4.4.1 接口定义

本接口是为应用配置支付渠道参数，前边为应用绑定了服务类型，此接口即为应用所绑定的服务类型配置支付渠道参数。

1、接口描述：保存支付渠道参数

2、接口定义如下：

在PayChannelService中定义createPayChannelParam方法：

```
/**
 * 保存支付渠道参数
 * @param payChannelParam 商户原始支付渠道参数
 */
void savePayChannelParam(PayChannelParamDTO payChannelParam) throws BusinessException;
```

5.4.4.2 接口实现

服务层提供一个接口实现支付渠道参数配置，如果该应用的服务类型已经配置某支付渠道参数则执行更新操作，否执行添加操作。

例如：

在PayChannelServiceImpl类中定义savePayChannelParam实现方法：

```
@Autowired
private PayChannelParamMapper payChannelParamMapper;

/**
 * 保存支付渠道参数
```



```
* @param payChannelParamDTO 支付渠道参数
* @throws BusinessException
*/
@Override
public void savePayChannelParam(PayChannelParamDTO payChannelParamDTO) throws
BusinessException {
    if(payChannelParamDTO == null || StringUtils.isBlank(payChannelParamDTO.getAppId())
        ||
        StringUtils.isBlank(payChannelParamDTO.getPlatformChannelCode())
        ||
        StringUtils.isBlank(payChannelParamDTO.getPayChannel())){
        throw new BusinessException(CommonErrorCode.E_300009);
    }
    //根据appid和服务类型查询应用与服务类型绑定id
    Long appPlatformChannelId = selectIdByAppPlatformChannel(payChannelParamDTO.getAppId(),
payChannelParamDTO.getPlatformChannelCode());
    if(appPlatformChannelId == null){
        //应用未绑定该服务类型不可进行支付渠道参数配置
        throw new BusinessException(CommonErrorCode.E_300010);
    }
    //根据应用与服务类型绑定id和支付渠道查询参数信息
    PayChannelParam payChannelParam = payChannelParamMapper.selectOne(new
LambdaQueryWrapper<PayChannelParam>().eq(PayChannelParam::getAppPlatformChannelId,
appPlatformChannelId)
        .eq(PayChannelParam::getPayChannel, payChannelParamDTO.getPayChannel()));

    //更新已有配置
    if (payChannelParam!=null){
        payChannelParam.setChannelName(payChannelParamDTO.getChannelName());
        payChannelParam.setParam(payChannelParamDTO.getParam());
        payChannelParamMapper.updateById(payChannelParam);
    }else{
        //添加新配置
        PayChannelParam entity =
PayChannelParamConvert.INSTANCE.dto2entity(payChannelParamDTO);
        entity.setId(null);
        //应用与服务类型绑定id
        entity.setAppPlatformChannelId(appPlatformChannelId);
        payChannelParamMapper.insert(entity);
    }
}

/**
 * 根据appid和服务类型查询应用与服务类型绑定id
 * @param appId
 * @param platformChannelCode
 * @return
 */
private Long selectIdByAppPlatformChannel(String appId,String platformChannelCode){
    //根据appid和服务类型查询应用与服务类型绑定id
    AppPlatformChannel appPlatformChannel = appPlatformChannelMapper.selectOne(new
LambdaQueryWrapper<AppPlatformChannel>().eq(AppPlatformChannel::getAppId, appId)

        .eq(AppPlatformChannel::getPlatformChannel, platformChannelCode));
```

```
        if(appPlatformChannel!=null){  
            return appPlatformChannel.getId();  
        }  
        return null;  
    }  
}
```

5.4.5 商户平台应用支付渠道参数配置接口(接口⑧)

5.4.5.1 接口定义

1、接口描述：请求交易服务保存支付渠道参数配置

2、接口定义如下：

前端提供两个接口：新增和更新

在PlatformParamController类中下定义createPayChannelParam

```
@ApiOperation("商户配置支付渠道参数")  
@ApiImplicitParams({  
    @ApiImplicitParam(name = "payChannelParam", value = "商户配置支付渠道参数", required =  
true, dataType = "PayChannelParamDTO", paramType = "body")  
})  
@RequestMapping(value = "/my/pay-channel-params", method =  
{RequestMethod.POST, RequestMethod.PUT})  
public void createPayChannelParam(@RequestBody PayChannelParamDTO payChannelParam){  
    Long merchantId = SecurityUtil.getMerchantId();  
    payChannelParam.setMerchantId(merchantId);  
    payChannelService.savePayChannelParam(payChannelParam);  
}
```

5.4.5.2 接口测试

微信：

1. 微信C扫B渠道参数配置，其中请求参数param和payChannel使用下面的配置

```
{  
    "appID": "wxd2bf2dba2e86a8c7",  
    "appSecret": "cec1a9185ad435abe1bced4b93f7ef2e",  
    "key": "95fe355daca50f1ae82f0865c2ce87c8",  
    "mchID": "1502570431",  
    "payKey": "95fe355daca50f1ae82f0865c2ce87c8",  
}  
"payChannel": "WX_JSAPI"
```

支付宝：

1. 申请支付宝开放平台账号，并获取支付宝渠道参数信息

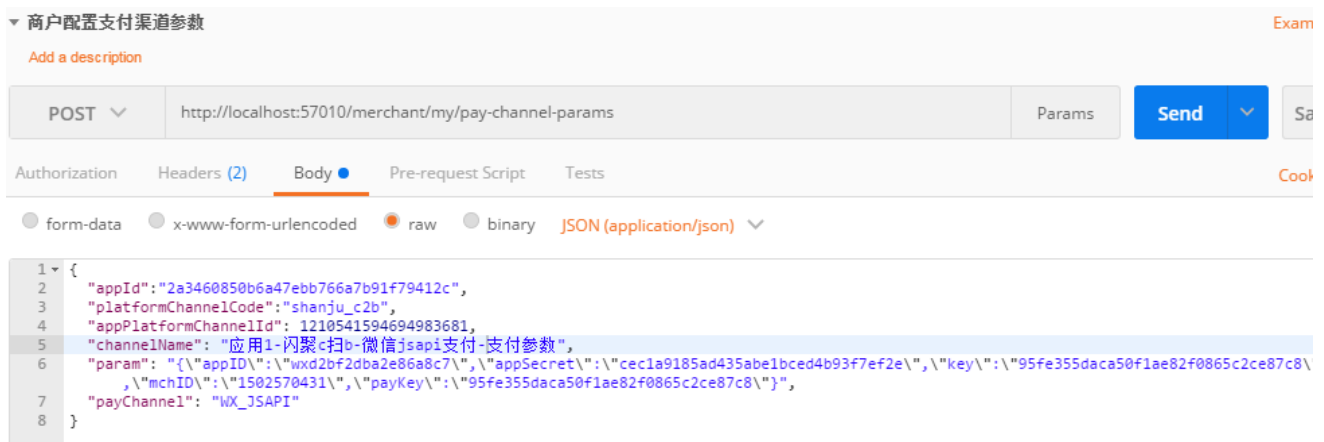
详见"支付宝开放平台使用指南.pdf"

2. 支付宝C扫B渠道参数配置，将其中的appId、rsaPrivateKey、alipayPublicKey、notifyUrl、returnUrl替换为自己的配置。

```
{
  "appId": "2016101000652290",
  "rsaPrivateKey": "MIIEvQIBADANBgkqhkiG9w0...",
  "alipayPublicKey": "MIIBIjANBgkqhkiG9w0BAQ...",
  "notifyUrl": "http://127.0.0.1:56010/payment-receiver/alipay-notify",
  "returnUrl": "http://127.0.0.1:56010/payment-receiver/alipay-return",

  "url": "https://openapi.alipaydev.com/gateway.do",
  "charset": "UTF-8",
  "format": "json",
  "log_path": "/log",
  "signtype": "RSA2"
}

"payChannel": "ALIPAY_WAP"
```



上图中，param为json串，需要使用EditPlus软件将json串合并为一行，并且用idea工具将字符串双引号转义。



然后替换掉制表符。

最后使用idea将双引号转义，如下：

```
"{ \"appID\": \"wxd2bf2dba2e86a8c7\", \"appSecret\": \"cec1a9185ad435abe1bcd4b93f7ef2e\",  
  \"key\": \"95fe355daca50f1ae82f0865c2ce87c8\", \"mchID\": \"1502570431\", \"payKey\":  
  \"95fe355daca50f1ae82f0865c2ce87c8\" }"
```

5.5 支付渠道参数查询

5.5.1 交易服务渠道参数查询接口

5.5.1.1 接口定义1

1、接口描述

1) 根据应用和服务类型获取原始支付参数param，结果可能是多个(支付宝param 微信param)

2、接口定义如下：

在PayChannelService接口中定义queryPayChannelParamByAppAndPlatform方法：

```
/**  
 * 获取指定应用指定服务类型下所包含的原始支付渠道参数列表  
 * @param appId 应用id  
 * @param platformChannel 服务类型  
 * @return  
 */  
List<PayChannelParamDTO> queryPayChannelParamByAppAndPlatform(String appId, String  
platformChannel)  
    throws BusinessException;
```

5.5.1.2 接口定义2

1、接口描述

1) 根据应用、服务类型及支付渠道获取原始支付参数param，结果只能是一个

2、接口定义如下：

在PayChannelService接口中定义queryParamByAppPlatformAndPayChannel方法：

```
/**
 * 获取指定应用指定服务类型下所包含的某个原始支付参数
 * @param appId
 * @param platformChannel
 * @param payChannel
 * @return
 * @throws BusinessException
 */
PayChannelParamDTO queryParamByAppPlatformAndPayChannel(String appId, String platformChannel,
String payChannel) throws BusinessException;
```

5.5.1.2 接口实现

在PayChannelServiceImpl中实现queryPayChannelParamByAppAndPlatform :

```
/**
 * 查询支付渠道参数
 *
 * @param appId          应用id
 * @param platformChannel 服务类型代码
 * @return
 */
@Override
public List<PayChannelParamDTO> queryPayChannelParamByAppAndPlatform(String appId, String
platformChannel) {
    //查出应用id和服务类型代码在app_platform_channel主键
    Long appPlatformChannelId = selectIdByAppPlatformChannel(appId,platformChannel);
    //根据appPlatformChannelId从pay_channel_param查询所有支付参数
    List<PayChannelParam> payChannelParams = payChannelParamMapper.selectList(new
LambdaQueryWrapper<PayChannelParam>().eq(PayChannelParam::getAppPlatformChannelId,
appPlatformChannelId));
    return PayChannelParamConvert.INSTANCE.listentity2listdto(payChannelParams);
}
```

在PayChannelServiceImpl中实现queryParamByAppPlatformAndPayChannel :


```
@Override
public PayChannelParamDTO queryParamByAppPlatformAndPayChannel(String appId, String
platformChannel,
                                String payChannel) throws
BusinessException {
    List<PayChannelParamDTO> payChannelParamDTOS = queryPayChannelParamByAppAndPlatform(appId,
platformChannel);
    for(PayChannelParamDTO payChannelParam:payChannelParamDTOS){
        if(payChannelParam.getPayChannel().equals(payChannel)){
            return payChannelParam;
        }
    }
    return null;
}
```

5.5.2 商户平台应用渠道参数查询接口

5.5.2.1 接口定义1

1、接口描述：根据应用和服务类型获取原始支付参数列表

2、接口定义如下：

在PlatformParamController中定义queryPayChannelParam

```
@ApiOperation("获取指定应用指定服务类型下所包含的原始支付渠道参数列表")
@ApiImplicitParams({
    @ApiImplicitParam(name = "appId", value = "应用id", required = true, dataType =
"String", paramType = "path"),
    @ApiImplicitParam(name = "platformChannel", value = "服务类型", required = true,
dataType = "String", paramType = "path")})
@GetMapping(value = "/my/pay-channel-params/apps/{appId}/platform-
channels/{platformChannel}")
public List<PayChannelParamDTO> queryPayChannelParam(@PathVariable String appId,
    @PathVariable String platformChannel) {
    return payChannelService.queryPayChannelParamByAppAndPlatform(appId, platformChannel);
}
```

5.5.2.1 接口定义2

1、接口描述：根据应用、服务类型及支付渠道获取原始支付参数

2、接口定义如下：

在PlatformParamController中定义queryPayChannelParam

```
@ApiOperation("获取指定应用指定服务类型下所包含的某个原始支付参数")
@ApiImplicitParams({
    @ApiImplicitParam(name = "appId", value = "应用id", required = true, dataType = "String",
        paramType = "path"),
    @ApiImplicitParam(name = "platformChannel", value = "平台支付渠道编码", required = true,
        dataType = "String", paramType = "path"),
    @ApiImplicitParam(name = "payChannel", value = "实际支付渠道编码", required = true,
        dataType = "String", paramType = "path")})
@GetMapping(value = "/my/pay-channel-params/apps/{appId}/platform-
channels/{platformChannel}/pay-channels/{payChannel}")
public PayChannelParamDTO queryPayChannelParam(@PathVariable String appId,
    @PathVariable String platformChannel, @PathVariable String payChannel) {
    return payChannelService.queryParamByAppPlatformAndPayChannel(appId, platformChannel,
        payChannel);
}
```

5.5.2.2 接口测试

postman : get:<http://localhost:57010/merchant/my/pay-channel-params/apps/{appId}/platform-channels/{platformChannel}>

5.6 支付渠道参数缓存

5.6.1 需求分析

渠道参数查询频繁，每一次支付都会查询渠道参数，为提供查询性能这里我们将渠道参数缓存到redis中，缓存流程如下：

1、保存渠道参数添加缓存

保存渠道参数成功，同时将渠道参数保存在Redis中。

2、查询渠道参数缓存

查询渠道参数，先从Redis查询，如果Redis存在则返回渠道参数，否则从数据库查询同时将查询到的渠道参数存储在Redis中。

5.6.2 搭建redis环境

1、安装redis

参考 资料下的“Redis安装指南.pdf”。

2、在Nacos中添加Redis配置：spring-boot-redis.yaml，Group：COMMON_GROUP

```
spring:
```

```
redis:
  # Redis数据库索引(默认为0)
  database: 0
  host: 127.0.0.1
  port: 6379
  # 连接超时时间(毫秒)
  timeout: 1000ms
  password: 123456
  lettuce:
    pool:
      # 连接池中的最大空闲连接
      max-idle: 8
      # 连接池中的最小空闲连接
      min-idle: 0
      # 连接池最大连接数(使用负值表示没有限制)
      max-active: 8
      # 连接池最大阻塞等待时间(使用负值表示没有限制)
      max-wait: 1000ms
      shutdown-timeout: 1000ms
```

在shanjupay-transaction-service工程的bootstrap.yml中引入Redis配置

```
-
  refresh: true
  data-id: spring-boot-redis.yaml # redis配置
  group: COMMON_GROUP # 通用配置组
```

3、添加Redis的pom依赖，shanjupay-transaction-service工程的pom.xml：

```
<!-- Redis启动器 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

4、复制RedisEnum, RedisUtil到shanjupay-common工程

5、复制RedisCache到shanjupay-transaction-service工程的com.shanjupay.transaction.common.util包下

5、添加Redis配置文件，包：com.shanjupay.transaction.config

```
package com.shanjupay.transaction.config;

import com.shanjupay.common.cache.Cache;
import com.shanjupay.transaction.common.util.RedisCache;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.core.StringRedisTemplate;

/**
 * redis 的配置类
```

```
*
*/
@Configuration
public class RedisConfig {

    @Bean
    public Cache cache(StringRedisTemplate redisTemplate){
        return new RedisCache(redisTemplate);
    }

}
```

5.6.3 保存缓存

保存渠道参数成功，同时将渠道参数保存在Redis中。

修改交易服务保存支付渠道参数接口,修改PayChannelServiceImpl中的savePayChannelParam方法：

```
@Resource
private Cache cache;

@Override
public void savePayChannelParam(PayChannelParamDTO payChannelParamDTO) throws
BusinessException {
    ...

    //更新缓存
    updateCache(payChannelParamDTO.getAppId(),payChannelParamDTO.getPlatformChannelCode());
}

private void updateCache(String appId, String platformChannel) {
    //处理redis缓存
    //1.key的构建 如：SJ_PAY_PARAM:b910da455bc84514b324656e1088320b:shanju_c2b
    String redisKey = RedisUtil.keyBuilder(appId, platformChannel);
    //2.查询redis,检查key是否存在
    Boolean exists = cache.exists(redisKey);
    if (exists) { //存在，则清除
        //删除原有缓存
        cache.del(redisKey);
    }
    //3.从数据库查询应用的服务类型对应的实际支付参数，并重新存入缓存
    List<PayChannelParamDTO> paramDTOS =
queryPayChannelParamByAppAndPlatform(appId,platformChannel);
    if (paramDTOS != null) {
        //存入缓存
        cache.set(redisKey, JSON.toJSON(paramDTOS).toString());
    }
}
```

5.6.4 查询缓存

查询渠道参数，先从Redis查询，如果Redis存在则返回渠道参数，否则从数据库查询同时将查询到的渠道参数存储在Redis中。

修改PayChannelServiceImpl中的queryPayChannelParamByAppAndPlatform方法：

```
/**
 * 查询支付渠道参数
 *
 * @param appId          应用id
 * @param platformChannel 服务类型代码
 * @return
 */
@Override
public List<PayChannelParamDTO> queryPayChannelParamByAppAndPlatform(String appId, String platformChannel) {
    //从缓存查询
    //1.key的构建 如：SJ_PAY_PARAM:b910da455bc84514b324656e1088320b:shanju_c2b
    String redisKey = RedisUtil.keyBuilder(appId, platformChannel);
    //是否有缓存
    Boolean exists = cache.exists(redisKey);
    if(exists){
        //从redis获取key对应的value
        String value = cache.get(redisKey);
        //将value转成对象
        List<PayChannelParamDTO> paramDTOS = JSONObject.parseArray(value, PayChannelParamDTO.class);
        return paramDTOS;
    }
    //查出应用id和服务类型代码在app_platform_channel主键
    Long appPlatformChannelId = selectIdByAppPlatformChannel(appId, platformChannel);
    //根据appPlatformChannelId从pay_channel_param查询所有支付参数
    List<PayChannelParam> payChannelParams = payChannelParamMapper.selectList(new LambdaQueryWrapper<PayChannelParam>().eq(PayChannelParam::getAppPlatformChannelId, appPlatformChannelId));
    List<PayChannelParamDTO> paramDTOS = PayChannelParamConvert.INSTANCE.listentity2listdto(payChannelParams);
    //存入缓存
    updateCache(appId, platformChannel);
    return paramDTOS;
}
```