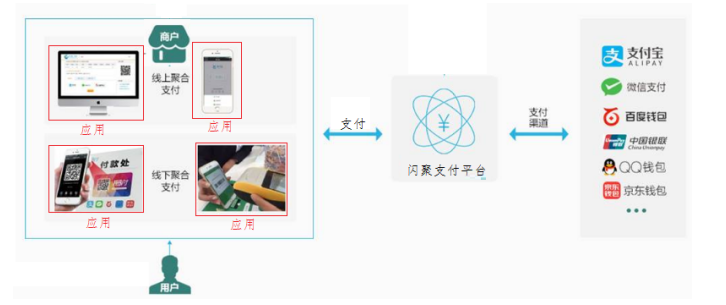
## 1.需求概述

### 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宜亩支付平台提供的服务，宜亩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：

宜亩支付平台提供**线上支付和线下支付**两个方式，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、宜亩支付平台对接微信、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宜亩支付平台

4、宜亩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。宜亩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？应用是商户在宜亩支付平台创建的业务标识，比如：商户在自己的XX电商网站使用宜亩支付则会创建“XX电商网站应用”，商户在自己经营的餐厅使用宜亩支付则会创建“XX餐厅应用”。

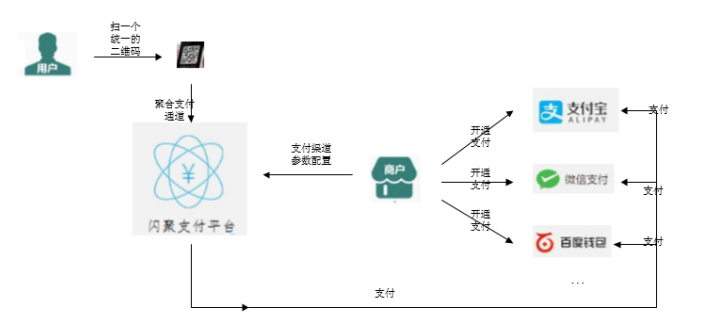
应用有什么用？用户是基于某个应用完成的支付，用户在商户的餐厅支付则支付订单会隶属于“XX餐厅应用”下，用户在XX电商网站支付则支付订单会隶属于“XX电商网站应用”下。

宜亩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

宜亩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，用户通过宜亩支付平台完成支付，宜亩支付平台最终会请求第三方支付渠道完成支付。

所以，商户不仅是宜亩平台的商户，还是第三方支付机构的商户，商户要使用宜亩支付平台就需要开通微信、支付宝等支付渠道，然后在宜亩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）



2、商户在宜亩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宜亩支付平台。

第一步在第三方支付渠道开通支付，商户将第三支付渠道的APPID、密钥等信息配置在宜亩支付平台。

3、宜亩支付平台为商户生成一个支付二维码

如果不使用宜亩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

服务类型是宜亩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

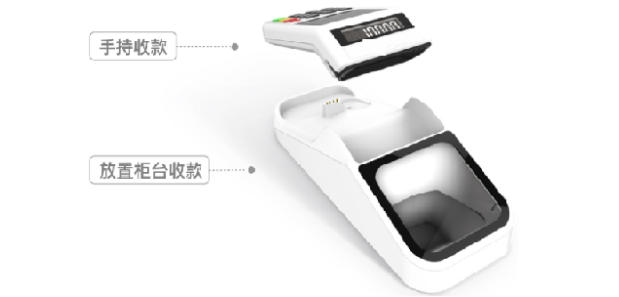
4）小程序支付等

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

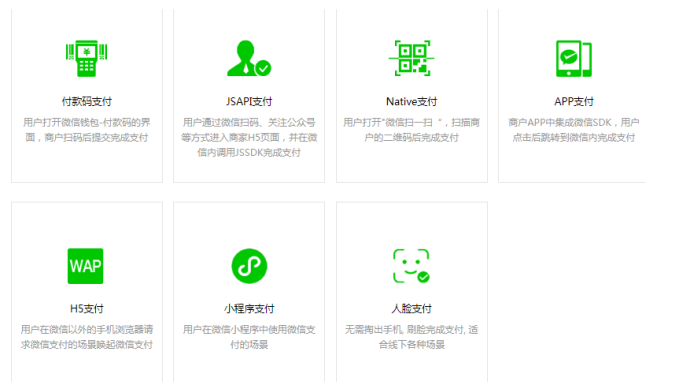


2、商户为应用绑定服务类型

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：



支付宝支付方式：



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，如果宜亩支付还聚合了百度、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。**

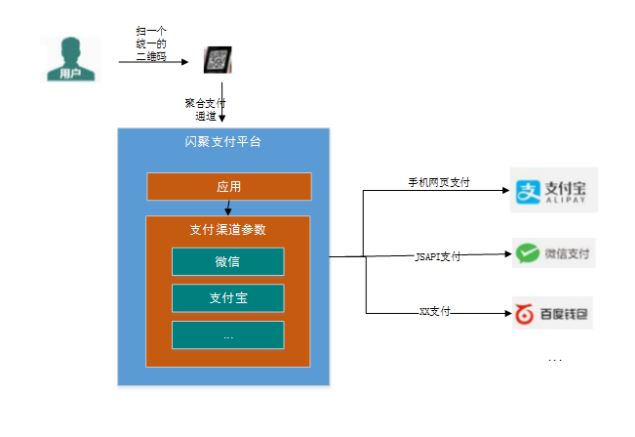
**三、总结**

下图展示了宜亩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。

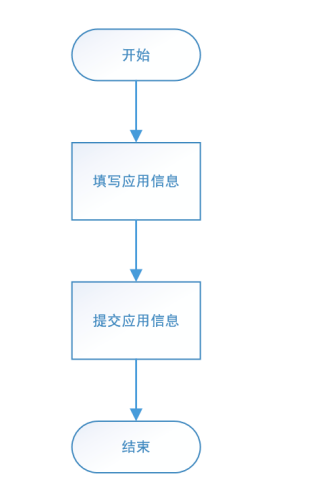
2、应用绑定宜亩支付平台提供的服务类型。

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 ：



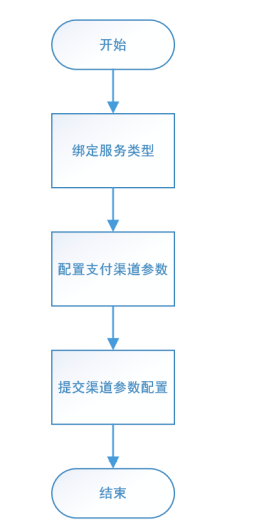
1. 填写应用基本信息



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



2）点击开启服务为应用绑定服务类型



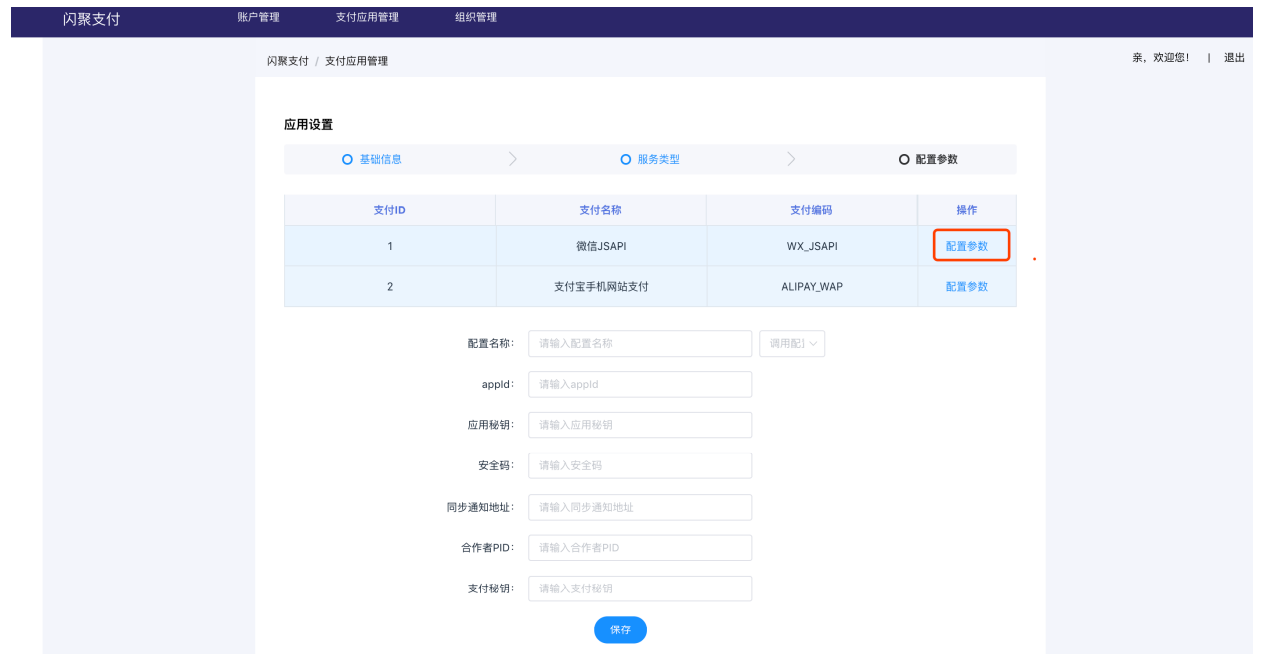
3）开启服务后，点击配置实际支付渠道按钮进入参数配置页面



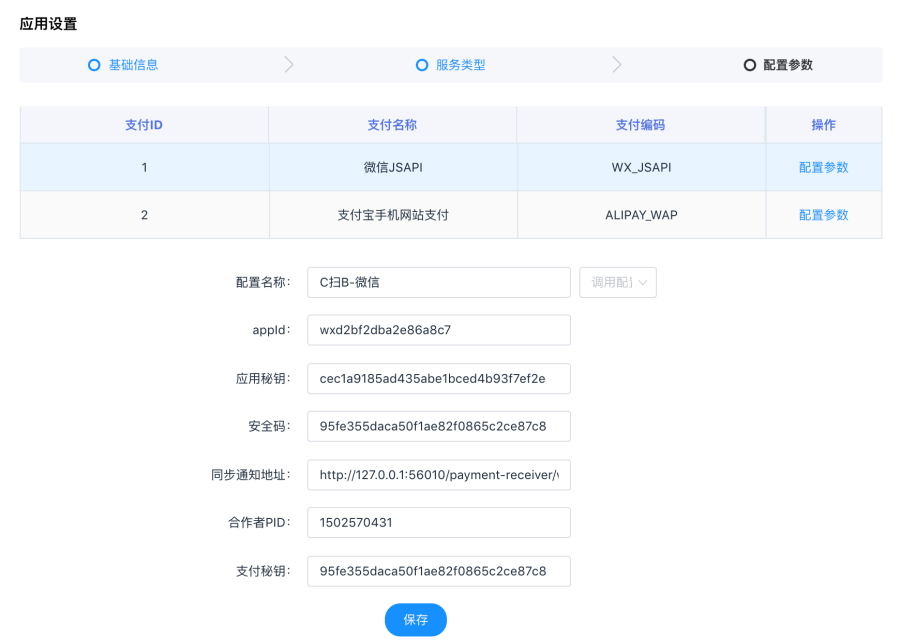
4）配置参数页面会显示对应服务类型下的原始支付渠道



5）点击配置参数按钮，为指定原始支付渠道配置



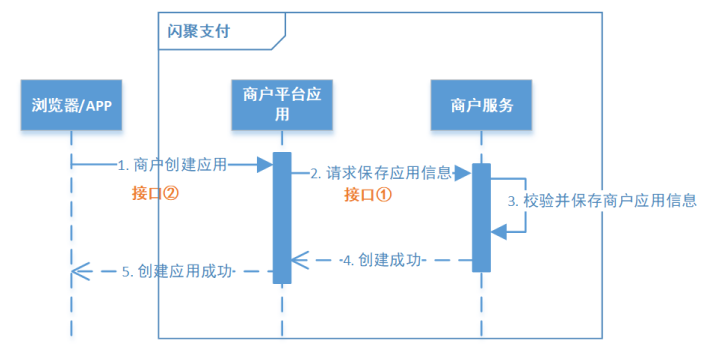
6）填写支付宝或微信的支付参数



## 2.商户应用创建

### 2.1.需求分析

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

### 2.2.应用创建

#### 2.2.1.商户服务创建应用接口（接口①）

##### 2.2.1.1.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：

|  |
| --- |
| package com.me.yimu.pay.merchant.api.dto;  import io.swagger.annotations.ApiModel;  import io.swagger.annotations.ApiModelProperty;  import lombok.Data;  import java.io.Serializable;  @Data  @ApiModel(value="AppDTO", description="")  public class AppDTO implements Serializable {  private static final long serialVersionUID = 1L;  private String appId;  @ApiModelProperty(value = "商店名称")  private String appName;  @ApiModelProperty(value = "所属商户")  private Long merchantId;  @ApiModelProperty(value = "应用公钥(RSAWithSHA256)")  private String publicKey;  @ApiModelProperty(value = "授权回调地址")  private String notifyUrl;  } |

在IAppService下定义createApp接口：

|  |
| --- |
| package com.me.yimu.pay.merchant.service;  import com.me.yimu.pay.domain.BusinessException;  import com.me.yimu.pay.merchant.api.dto.AppDTO;  import com.me.yimu.pay.merchant.entity.App;  import com.baomidou.mybatisplus.extension.service.IService;  /\*\*  \* <p>  \* 服务类  \* </p>  \*  \* @author author  \* @since 2022-12-05  \*/  public interface IAppService extends IService<App> {  /\*\*  \* 商户下创建应用  \* @return  \*/  AppDTO createApp(Long merchantId, AppDTO app) throws BusinessException;    } |

##### 2.2.1.2.接口实现

定义AppCovert负责对象转换。

|  |
| --- |
| package com.me.yimu.pay.merchant.convert;  import org.mapstruct.Mapper;  import org.mapstruct.factory.Mappers;  import com.me.yimu.pay.merchant.api.dto.AppDTO;  import com.me.yimu.pay.merchant.entity.App;  import java.util.List;  /\*\*  \* @author Administrator  \* @version 1.0  \*\*/  @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.me.yimu.pay.merchant.service.impl;  **import** com.me.yimu.pay.domain.BusinessException;  **import** com.me.yimu.pay.domain.CommonErrorCode;  **import** com.me.yimu.pay.merchant.api.dto.AppDTO;  **import** com.me.yimu.pay.merchant.convert.AppCovert;  **import** com.me.yimu.pay.merchant.entity.App;  **import** com.me.yimu.pay.merchant.entity.Merchant;  **import** com.me.yimu.pay.merchant.mapper.AppMapper;  **import** com.me.yimu.pay.merchant.mapper.MerchantMapper;  **import** com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  **import** com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  **import** com.me.yimu.pay.merchant.service.IAppService;  **import** com.me.yimu.pay.utils.RandomUuidUtil;  **import** lombok.extern.slf4j.Slf4j;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.transaction.annotation.Transactional;  **import** org.apache.dubbo.config.annotation.Service;  /\*\*  \* <p>  \* 服务实现类  \* </p>  \*  \* **@author** author  \* **@since** 2022-12-05  \*/  @Slf4j  @Service  @Transactional  **public** **class** AppServiceImpl **extends** ServiceImpl<AppMapper, App> **implements** IAppService {    @Autowired  **private** AppMapper appMapper;  @Autowired  **private** MerchantMapper merchantMapper;    **public** AppDTO createApp(Long merchantId, AppDTO app) **throws** BusinessException {  //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;  }  } |

#### 2.2.2.商户服务创建应用接口（接口②）

##### 2.2.2.1.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：

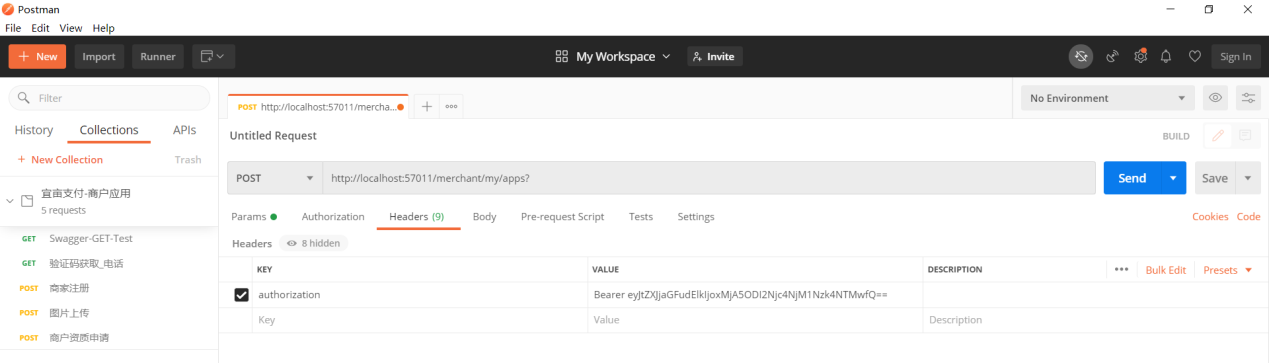
定义AppController类，并且定义createApp方法：

|  |
| --- |
| **package** com.me.yimu.pay.merchant.controller;  **import** org.springframework.web.bind.annotation.PostMapping;  **import** org.springframework.web.bind.annotation.RequestBody;  **import** org.springframework.web.bind.annotation.RestController;  **import** org.apache.dubbo.config.annotation.Reference;  **import** com.me.yimu.pay.merchant.api.AppService;  **import** com.me.yimu.pay.merchant.api.dto.AppDTO;  **import** com.me.yimu.pay.merchant.util.SecurityUtil;  **import** io.swagger.annotations.Api;  **import** io.swagger.annotations.ApiImplicitParam;  **import** io.swagger.annotations.ApiImplicitParams;  **import** io.swagger.annotations.ApiOperation;  /\*\*  \*  \* **@author** PC  \*  \*/  @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){  //得到商户id  Long merchantId = SecurityUtil.*getMerchantId*();  **return** appService.createApp(merchantId,app);  }  } |

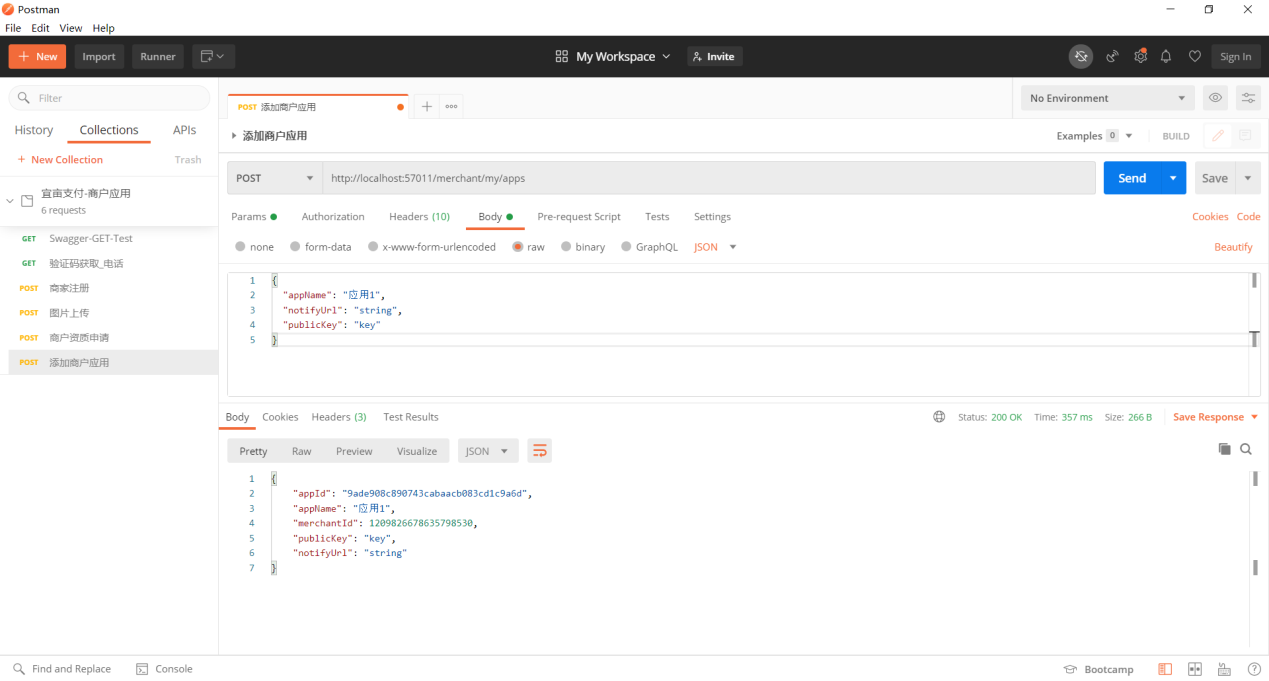
##### 2.2.2.2.接口测试

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。



2.添加应用



### 2.3.应用查询

#### 2.3.1.商户服务应用查询接口

##### 2.3.1.1.接口定义

1.根据商户ID查询应用

接口定义如下：AppService

|  |
| --- |
| /\*\*  \* 根据商户id查询应用列表  \* **@param** merchantId  \* **@return**  \* **@throws** BusinessException  \*/  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

|  |
| --- |
| /\*\*  \* 根据应用id查询应用信息  \* **@param** appId  \* **@return**  \* **@throws** BusinessException  \*/  AppDTO getAppById(String appId)**throws** BusinessException; |

##### 2.3.1.2.接口实现

AppServiceImpl.java

|  |
| --- |
| @Override  **public** List<AppDTO> queryAppByMerchant(Long merchantId) **throws** BusinessException {  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 appId) **throws** BusinessException {  App app = appMapper.selectOne(**new** QueryWrapper<App>().lambda().eq(App::getAppId, appId));  **return** AppCovert.***INSTANCE***.entity2dto(app);  } |

#### 2.3.2.商户平台应用查询接口

##### 2.3.2.1.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

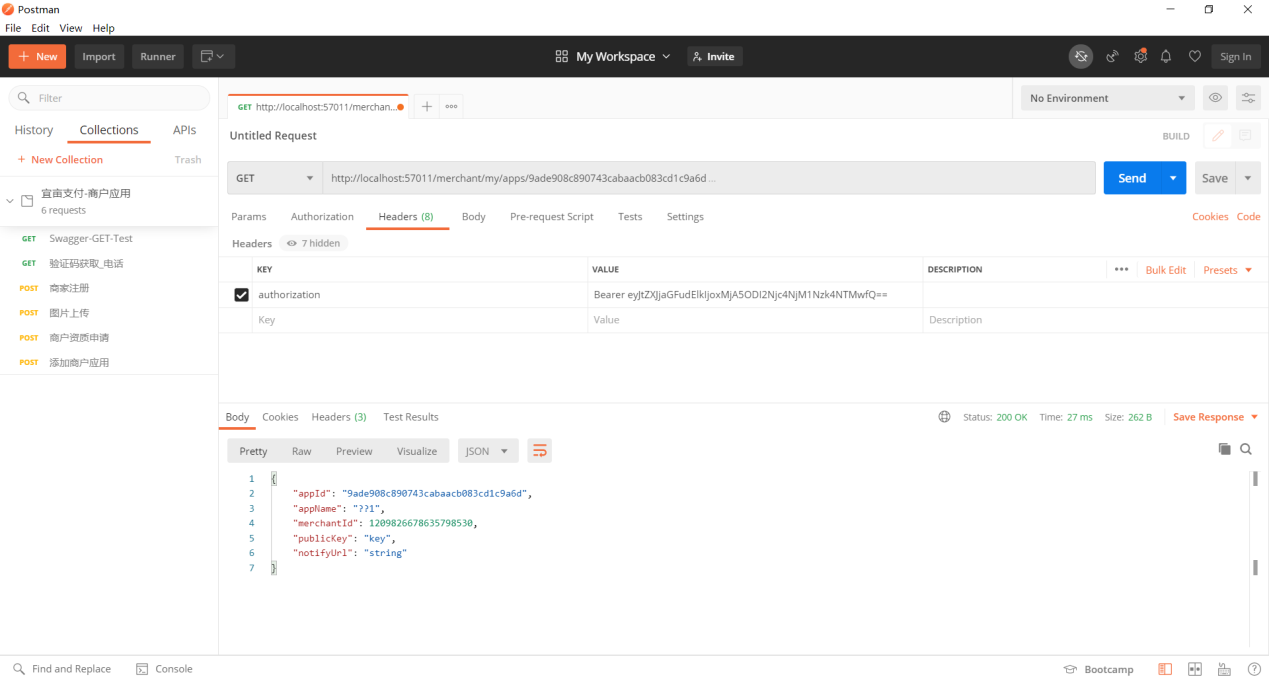
接口定义如下：AppController

|  |
| --- |
| @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);  } |

##### 2.3.2.2.接口测试

1.获取商户的应用列表

2.根据appId获取应用的详细信息

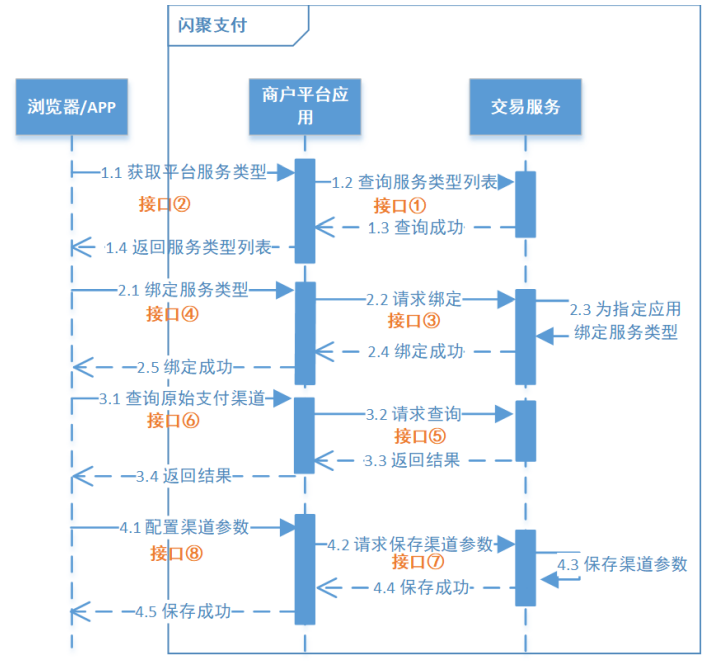


## 3.支付渠道参数配置

### 3.1.需求分析

#### 3.1.1.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



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

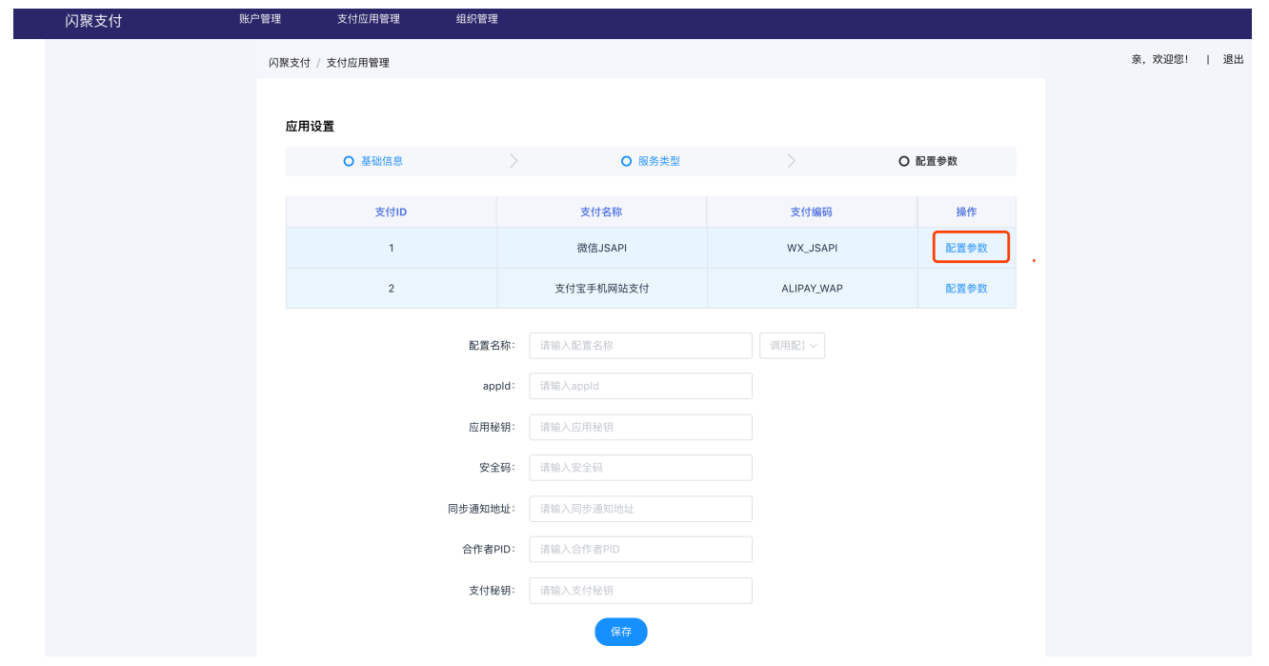


8. 请求交易服务获取列表

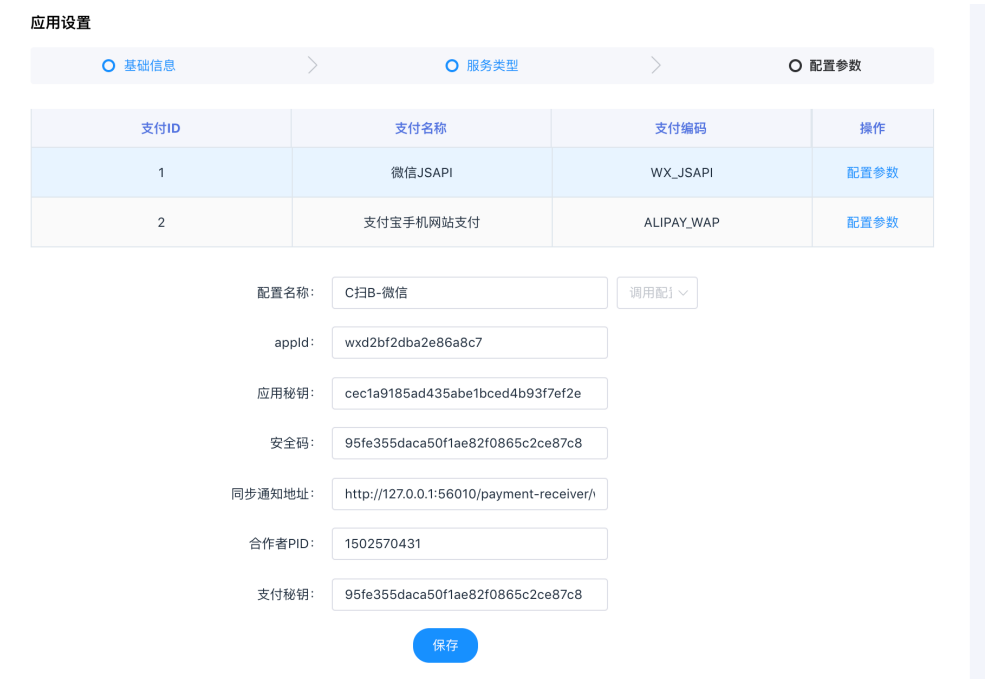
9. 返回结果给前端



10. 前端请求配置支付渠道参数



11. 商户平台应用请求交易服务保存参数配置



12. 返回前端保存成功

#### 3.1.2.基础数据

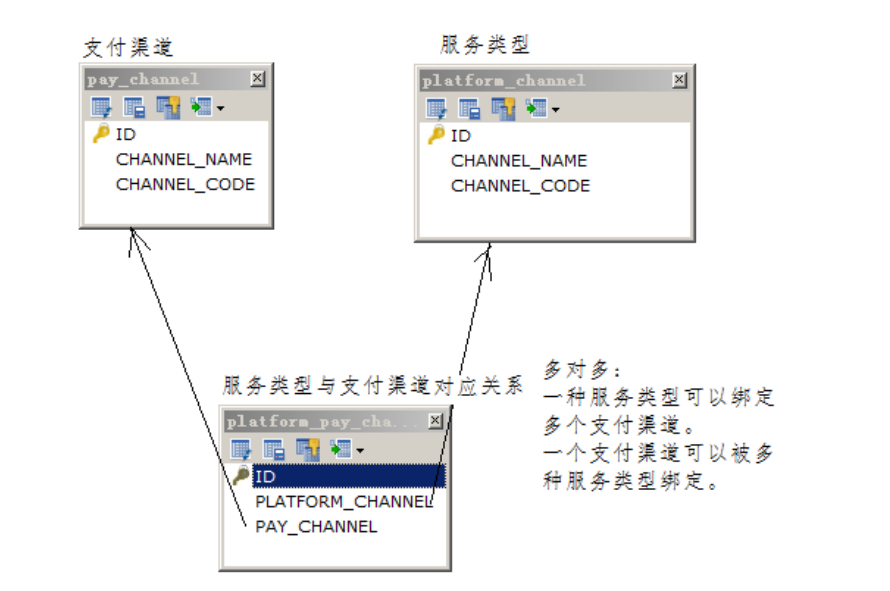
初始化平台基础数据：

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JSAPI接口和支付宝手机 网站支付接口。



|  |
| --- |
| use yimupay\_transaction;  LOCK TABLES `platform\_channel` WRITE;  /\*!40000 ALTER TABLE `platform\_channel` DISABLE KEYS \*/;  INSERT INTO `platform\_channel` (`ID`, `CHANNEL\_NAME`, `CHANNEL\_CODE`)  VALUES  (1,'宜亩B扫C','yimu\_b2c'),  (2,'宜亩C扫B','yimu\_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; |

### 3.2.搭建交易服务工程

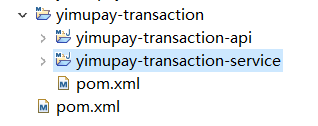
#### 3.2.1.交易服务介绍

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(yimupay-transaction-api) | 定义交易服务提供的接口 |
| 交易服务(yimupay-transaction-service) | 实现交易服务的接口实现 |

#### 3.2.2.搭建工程

1.创建工程



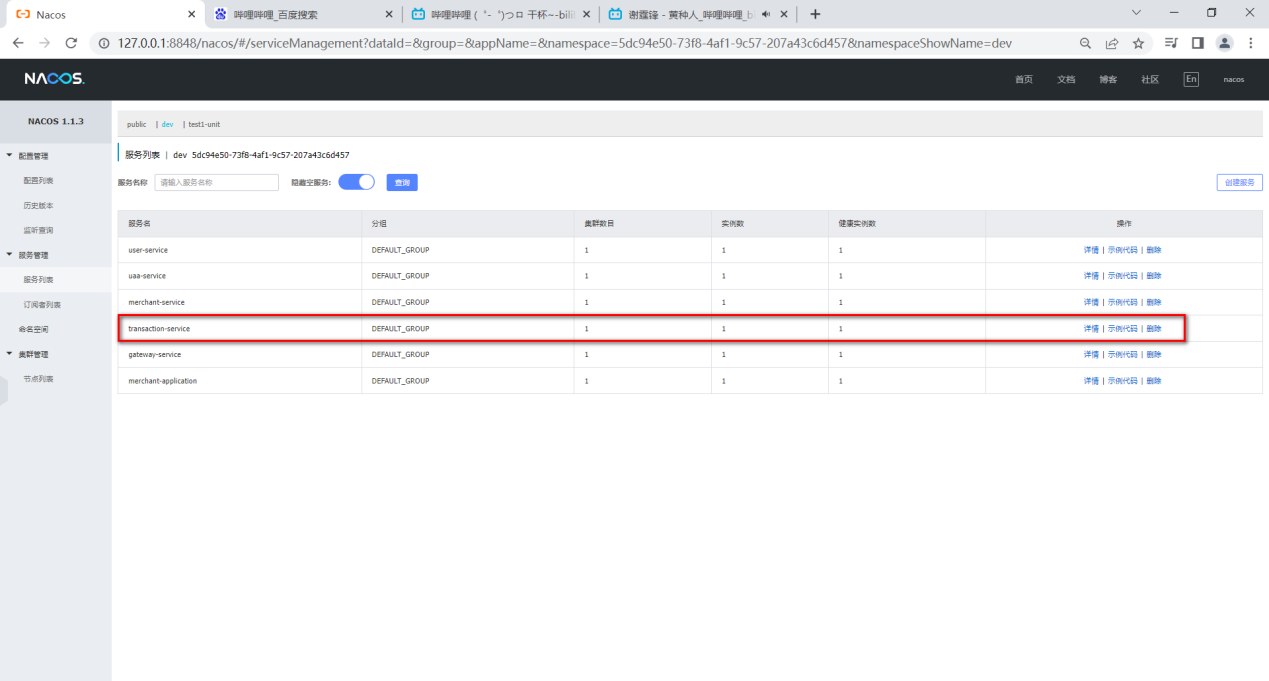
2.在Nacos中添加transaction-service.yaml配置，Group: YIMUPAY\_GROUP

|  |
| --- |
| # 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/yimupay\_transaction?useUnicode=true&serverTimezone=Asia/Shanghai&useSSL=false  username: root  password: mnj852123  # 覆盖spring‐boot‐mybatis‐plus.yaml的项目  mybatis-plus:  typeAliasesPackage: com.me.yimu.pay.transaction.entity  mapper-locations: classpath:com/yimupay/\*/mapper/\*.xml  #支付入口url  yimupay:  payurl: "http://127.0.0.1:56010/transaction/pay‐entry/" |

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

|  |
| --- |
| server:  port: 56050 #启动端口 命令行注入  nacos:  server:  addr: 127.0.0.1:8848  spring:  application:  name: transaction-service  main:  allow-bean-definition-overriding: **true** # Spring Boot 2.1 需要设定  cloud:  nacos:  discovery:  server-addr: ${nacos.server.addr}  **namespace: 5dc94e50-73f8-4af1-9c57-207a43c6d457**  cluster-name: DEFAULT  config:  server-addr: ${nacos.server.addr} # 配置中心地址  file-extension: yaml  **namespace: 5dc94e50-73f8-4af1-9c57-207a43c6d457 #2ed00aaa-b760-4171-baa9-83d92e509a41 # 默认开发环境郑州区 命令行注入**  group: YIMUPAY\_GROUP # 聚合支付业务组  ext-config:  -  refresh: **true**  data-id: spring-boot-http.yaml # spring boot http配置  group: COMMON\_GROUP # 通用配置组  -  refresh: **true**  data-id: spring-boot-starter-druid.yaml # spring boot starter druid配置  group: COMMON\_GROUP # 通用配置组  -  refresh: **true**  data-id: spring-boot-mybatis-plus.yaml # spring boot mybatisplus配置  group: COMMON\_GROUP # 通用配置组  - refresh: **true**  data-id: spring-boot-redis.yaml # redis配置  group: COMMON\_GROUP # 通用配置组  dubbo:  scan:  # dubbo 服务扫描基准包  base-packages: com.me  protocol:  # dubbo 协议  name: dubbo  port: 20893  registry:  address: nacos://127.0.0.1:8848  application:  qos:  port: 22250 # dubbo qos端口配置 命令行注入  consumer:  check: **false**  timeout: 3000  retries: -1  logging:  config: classpath:log4j2.xml |

4.启动TransactionBootstrap测试

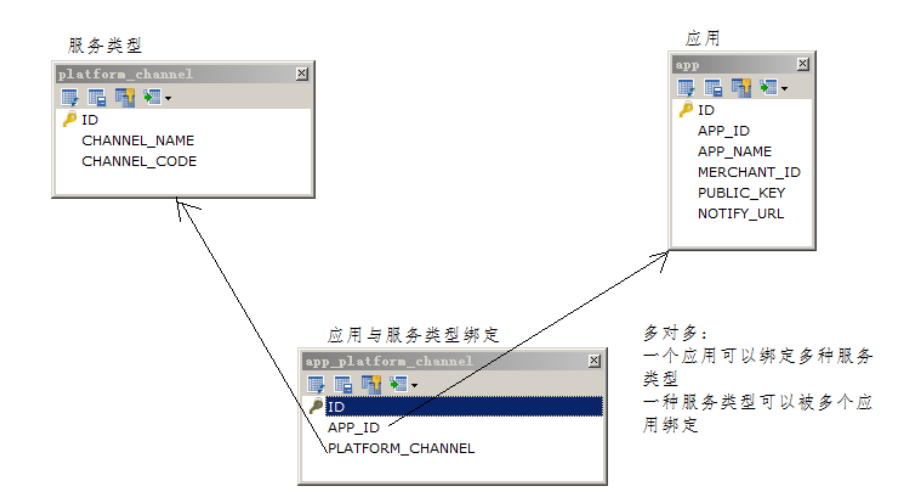


启动成功后观察nacos的服务列表，看transaction-service是否注册服务成功，如果注册成功则说明交易服务启动成功。

### 3.3.应用绑定服务类型

#### 3.3.1.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:



#### 3.3.2.交易服务获取平台服务类型(接口①)

绑定服务类型页面，页面中列表出服务类型。



##### 3.3.2.1.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.me.yimu.pay.transaction.api;  **import** java.util.List;  **import** com.me.yimu.pay.domain.BusinessException;  **import** com.me.yimu.pay.transaction.api.dto.PlatformChannelDTO;  /\*\*  \* 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  \* **@author** PC  \*  \*/  **public** **interface** PayChannelService {  /\*\*  \* 获取平台服务类型  \* **@return**  \* **@throws** BusinessException  \*/  List<PlatformChannelDTO> queryPlatformChannel() **throws** BusinessException;    } |

##### 3.3.2.2.接口实现

定义PayChannelServiceImpl类及queryPlatformChannel实现方法：

|  |
| --- |
| @Service  **public** **class** PayChannelServiceImpl **implements** PayChannelService {  @Autowired  **private** PlatformChannelMapper platformChannelMapper;    @Override  **public** List<PlatformChannelDTO> queryPlatformChannel() **throws** BusinessException {  List<PlatformChannel> platformChannels = platformChannelMapper.selectList(**null**);  List<PlatformChannelDTO> platformChannelDTOS =  PlatformChannelConvert.***INSTANCE***.listentity2listdto(platformChannels);  **return** platformChannelDTOS;  }  } |

#### 3.3.3.商户平台应用获取平台服务类型(接口②)

##### 3.3.3.1.接口定义

在商户平台应用工程添加依赖：

|  |
| --- |
| <dependency>  <groupId>com.me.yimu.pay.transaction.api</groupId>  <artifactId>yimupay-transaction-api</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency> |

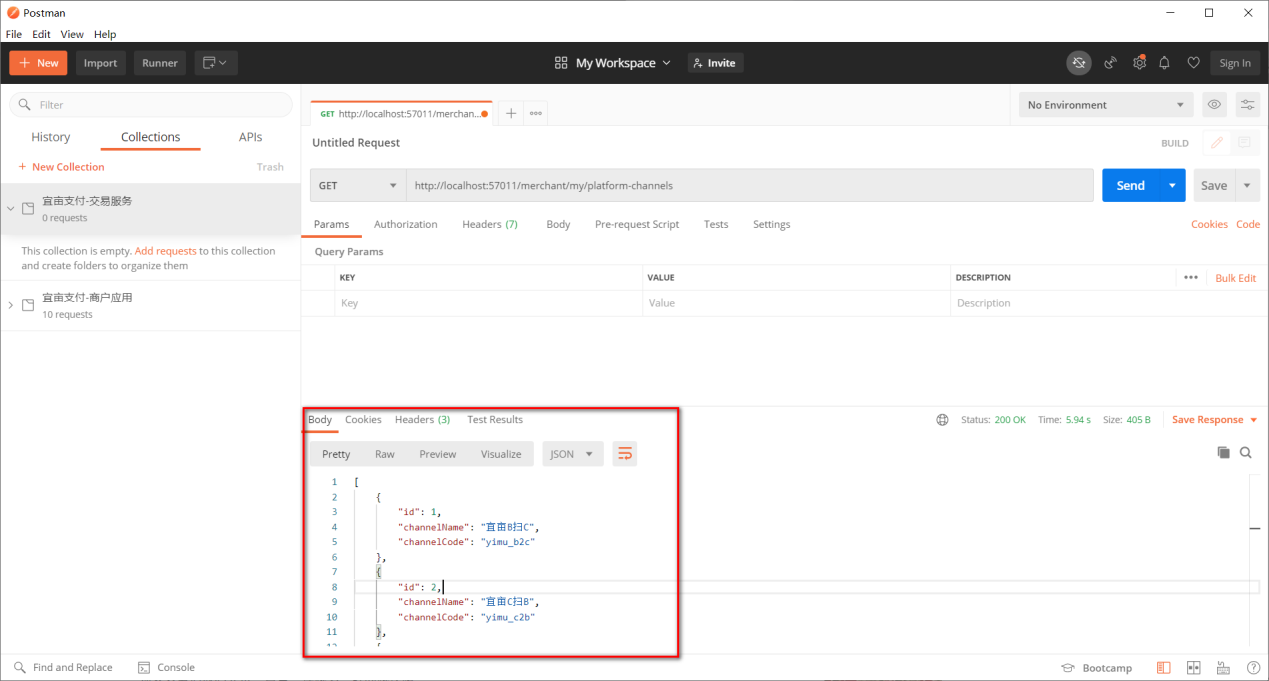
1、接口描述：请求交易服务查询平台支持的所有服务类型

2、接口定义如下：PlatformParamController

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

##### 3.3.3.2.接口测试

<http://localhost:57011/merchant/my/platform-channels>



#### 3.3.4.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。



##### 3.3.4.1.接口定义

1、接口描述：

1）查询出指定应用就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; |

##### 3.3.4.2.接口实现

定义PayChannelServiceImpl类及bindPlatformChannelForApp实现方法：

|  |
| --- |
| @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);  }  } |

#### 3.3.5.商户平台应用绑定服务类型接口(接口④)

##### 3.3.5.1.接口定义

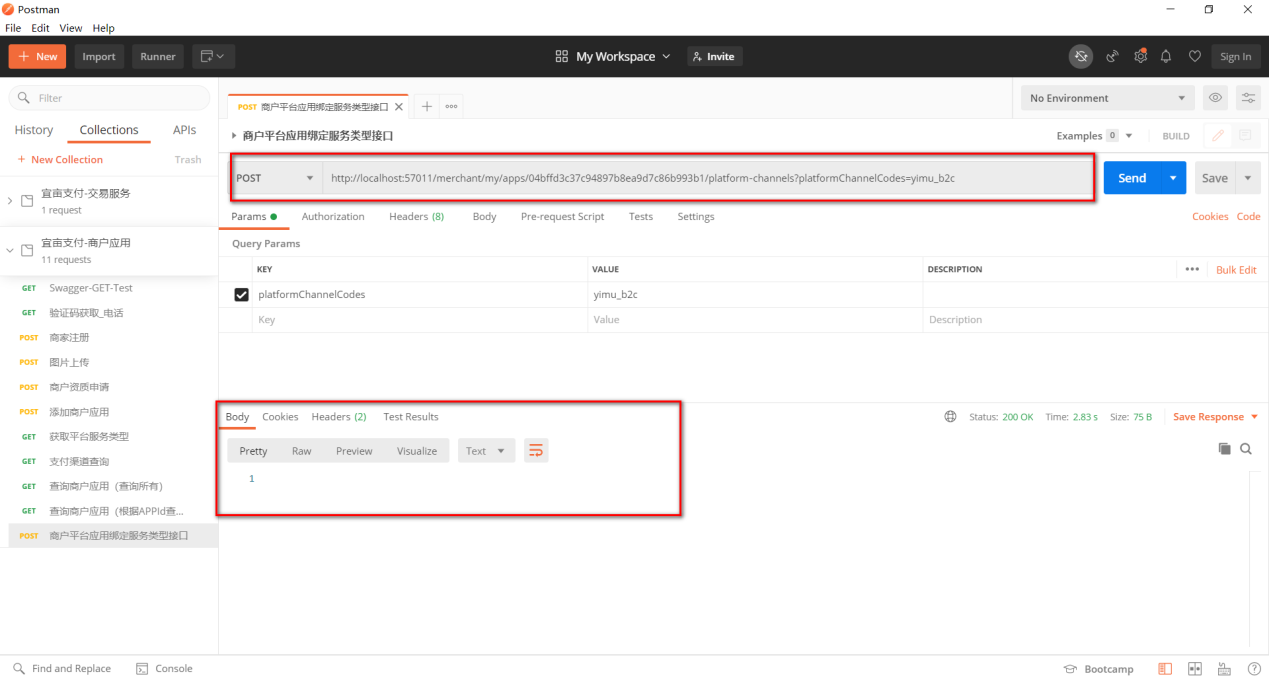
1、接口描述：请求交易服务为指定应用添加服务类型

2、接口定义如下： 在AppController类中定义bindPlatformForApp方法：

|  |
| --- |
| @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);  } |

##### 3.3.5.2.接口测试

http://localhost:57011/merchant/my/apps/04bffd3c37c94897b8ea9d7c86b993b1/platform‐channels?platformChannelCodes=yimu\_b2c



#### 3.3.6.交易服务查询服务类型绑定状态

##### 3.3.6.1.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; |

##### 3.3.6.2.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;  }  } |

#### 3.3.7.商户应用查询服务类型绑定状态

##### 3.3.7.1.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

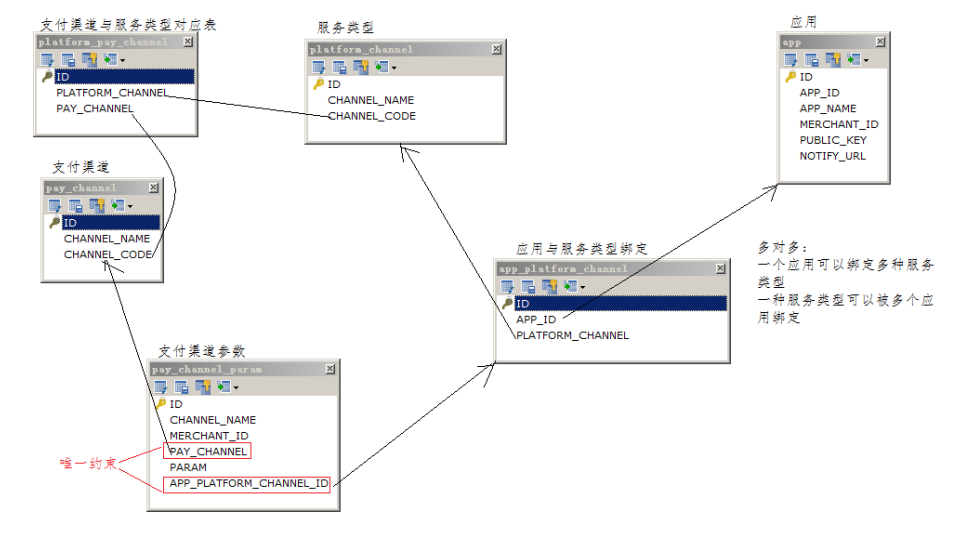
在AppController中定义：

|  |
| --- |
| @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);  } |

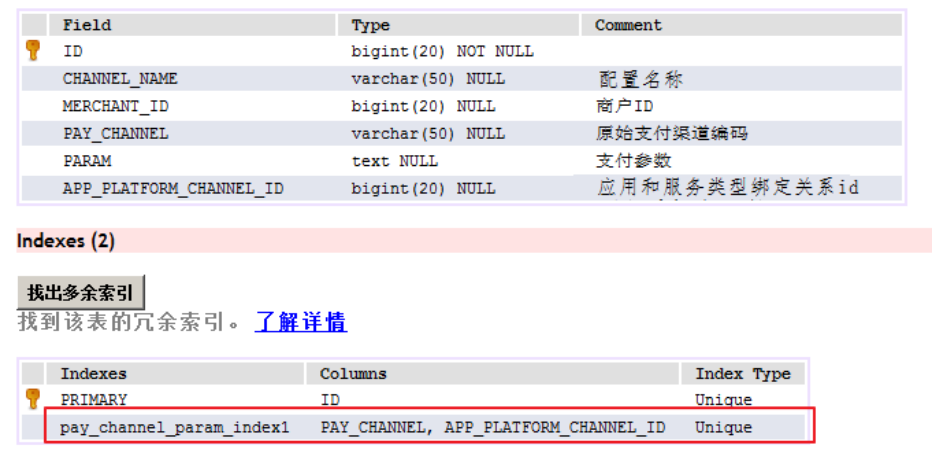
### 3.5.支付渠道参数配置

#### 3.5.1.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）



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。

#### 3.5.2.交易服务原始支付渠道查询接口(接口⑤)

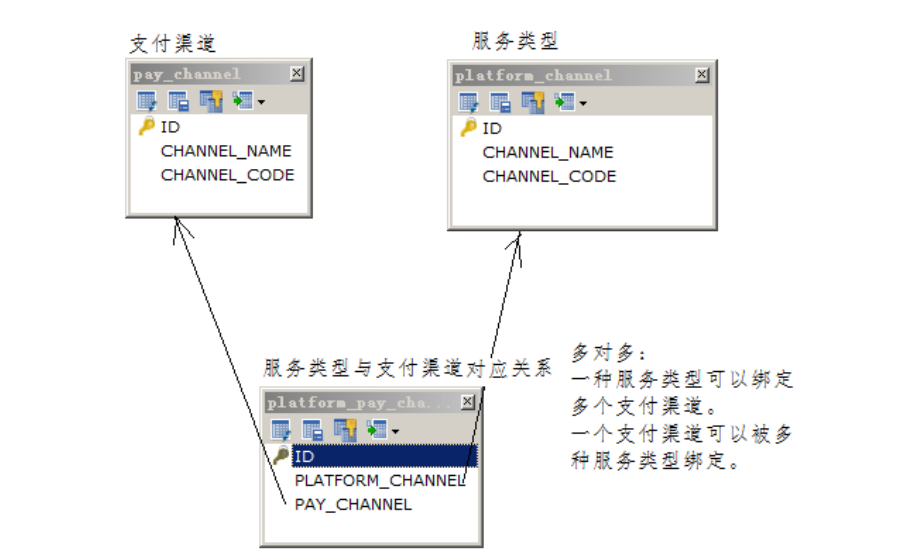
配置参数页面会显示对应服务类型下的原始支付渠道。



##### 3.5.2.1.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; |

##### 3.5.2.2.接口实现

1、在PlatformChannelMapper中定义selectPayChannelByPlatformChannel方法：

|  |
| --- |
| /\*\*  \* 根据服务类型code查询对应的支付渠道  \* **@param** platformChannelCode  \* **@return**  \*/  @Select("SELECT " +  " pc.\* " +  "FROM" +  " platform\_pay\_channel ppc," +  " pay\_channel pc," +  " platform\_channel pla " +  "WHERE ppc.PAY\_CHANNEL = pc.CHANNEL\_CODE " +  " AND ppc.PLATFORM\_CHANNEL = pla.CHANNEL\_CODE " +  " AND pla.CHANNEL\_CODE = #{platformChannelCode} ")  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);  } |

##### 3.5.2.3.接口测试

在交易 服务进行单元 测试，编写单元 测试类：

|  |
| --- |
| package com.me.yimu.pay.transaction.service.impl;  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 com.me.yimu.pay.transaction.api.PayChannelService;  import com.me.yimu.pay.transaction.api.dto.PayChannelDTO;  import java.util.List;  /\*\*  \* @author Administrator  \* @version 1.0  \*\*/  @SpringBootTest  @RunWith(SpringRunner.class)  @Slf4j  public class PayChannelServiceImpl {  @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);  }  } |

#### 3.5.3.商户平台应用支付渠道查询接口(接口⑥)

##### 3.5.3.1.接口定义

1、接口描述：根据服务类型查询支付渠道列表

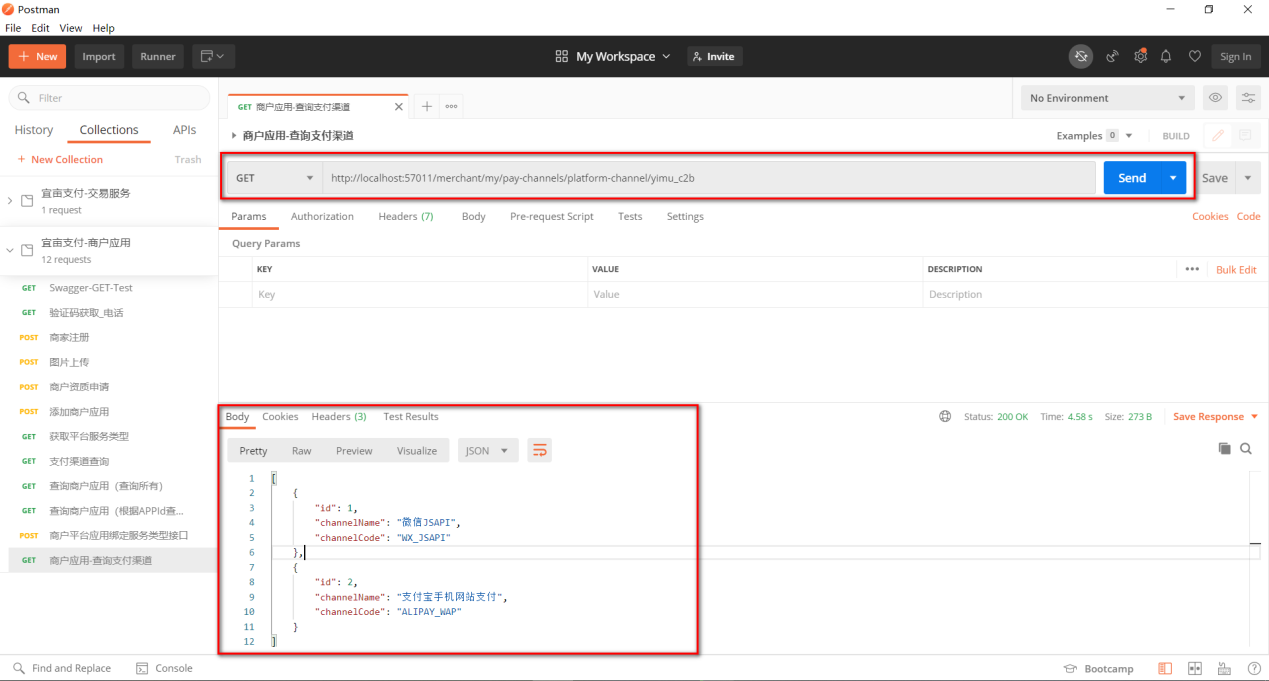
2、接口定义如下：

在PlatformParamController类中定义queryPayChannelByPlatformChannel：

|  |
| --- |
| @ApiOperation("根据服务类型查询支付渠道")  @ApiImplicitParam(name = "platformChannelCode", value = "服务类型代码", required = **true**, dataType = "String", paramType = "path")  @GetMapping(value="/my/pay-channels/platform-channel/{platformChannelCode}")  List<PayChannelDTO> queryPayChannelByPlatformChannel(@PathVariable("platformChannelCode") String platformChannelCode) **throws** BusinessException{  **return** payChannelService.queryPayChannelByPlatformChannel(platformChannelCode);  } |

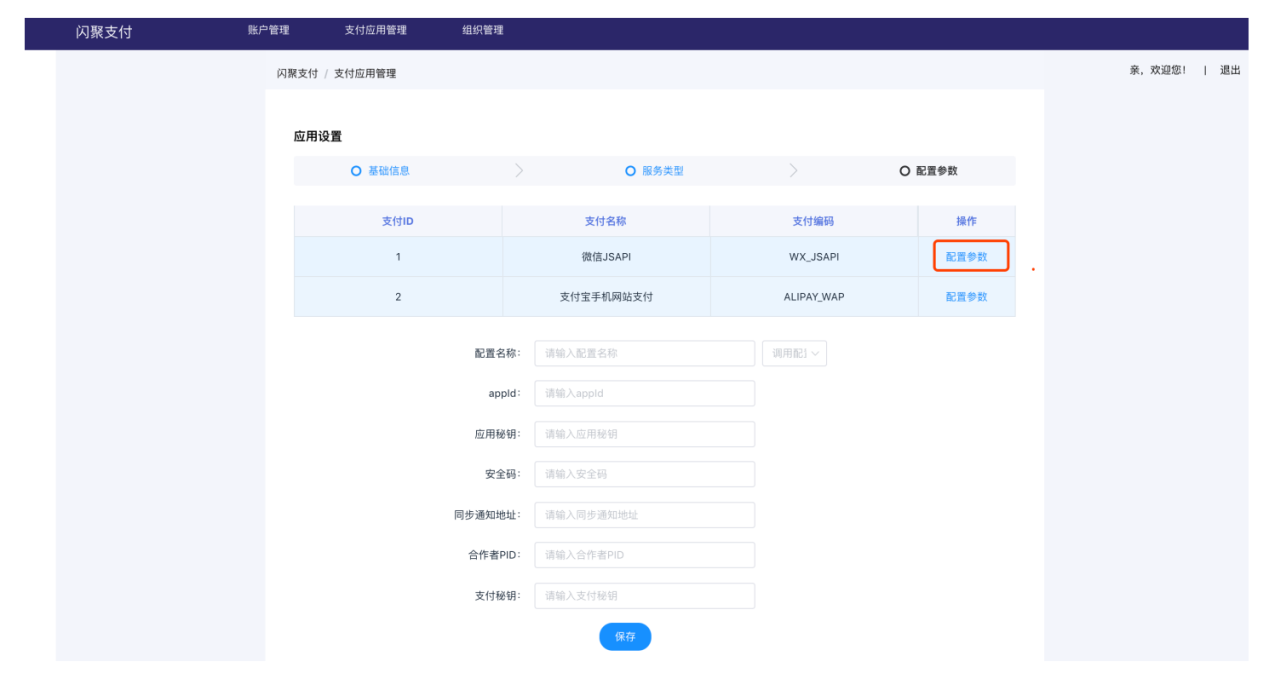
##### 3.5.3.2.接口测试

使用Postman：<http://localhost:57011/merchant/my/pay-channels/platform-channel/yimu_c2b>



#### 3.5.4.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。



##### 3.5.4.1.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savePayChannelParam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; |

##### 3.5.4.2.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：

|  |
| --- |
| /\*\*  \* 支付渠道参数配置  \*  \* **@param** payChannelParam 配置支付渠道参数：包括：商户id、应用id，服务类型code，支付渠道code，配置名称，配置参数(json)  \* **@throws** BusinessException  \*/  @Override  **public** **void** savePayChannelParam(PayChannelParamDTO payChannelParam) **throws** BusinessException {  **if**(payChannelParam == **null** || payChannelParam.getChannelName() == **null** || payChannelParam.getParam()== **null**){  **throw** **new** BusinessException(CommonErrorCode.***E\_300009***);  }  //根据应用、服务类型、支付渠道查询一条记录  //根据应用、服务类型查询应用与服务类型的绑定id  Long appPlatformChannelId = selectIdByAppPlatformChannel(payChannelParam.getAppId(), payChannelParam.getPlatformChannelCode());  **if**(appPlatformChannelId == **null**){  **throw** **new** BusinessException(CommonErrorCode.***E\_300010***);  }  //根据应用与服务类型的绑定id和支付渠道查询PayChannelParam的一条记录  PayChannelParam entity = payChannelParamMapper.selectOne(**new** LambdaQueryWrapper<PayChannelParam>().eq(PayChannelParam::getAppPlatformChannelId, appPlatformChannelId)  .eq(PayChannelParam::getPayChannel, payChannelParam.getPayChannel()));  //如果存在配置则更新  **if**(entity != **null**){  entity.setChannelName(payChannelParam.getChannelName());//配置名称  entity.setParam(payChannelParam.getParam());//json格式的参数  payChannelParamMapper.updateById(entity);  }**else**{  //否则添加配置  PayChannelParam entityNew = PayChannelParamConvert.***INSTANCE***.dto2entity(payChannelParam);  entityNew.setId(**null**);  entityNew.setAppPlatformChannelId(appPlatformChannelId);//应用与服务类型绑定关系id  payChannelParamMapper.insert(entityNew);  }  //保存到redis  updateCache(payChannelParam.getAppId(),payChannelParam.getPlatformChannelCode());  } |

#### 3.5.5.商户平台应用支付渠道参数配置接口(接口⑧)

##### 3.5.5.1.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")  @ApiImplicitParam(name = "payChannelParamDTO", value = "支付渠道参数", required = **true**, dataType = "PayChannelParamDTO", paramType = "body")  @RequestMapping(value = "/my/pay-channel-params",method = {RequestMethod.***POST***,RequestMethod.***PUT***})  **void** createPayChannelParam(@RequestBody PayChannelParamDTO payChannelParamDTO){  **if**(payChannelParamDTO == **null** || payChannelParamDTO.getChannelName() == **null**){  **throw** **new** BusinessException(CommonErrorCode.***E\_300009***);  }  //商户id  Long merchantId = SecurityUtil.*getMerchantId*();  payChannelParamDTO.setMerchantId(merchantId);  payChannelService.savePayChannelParam(payChannelParamDTO);  } |

##### 3.5.5.2.接口测试

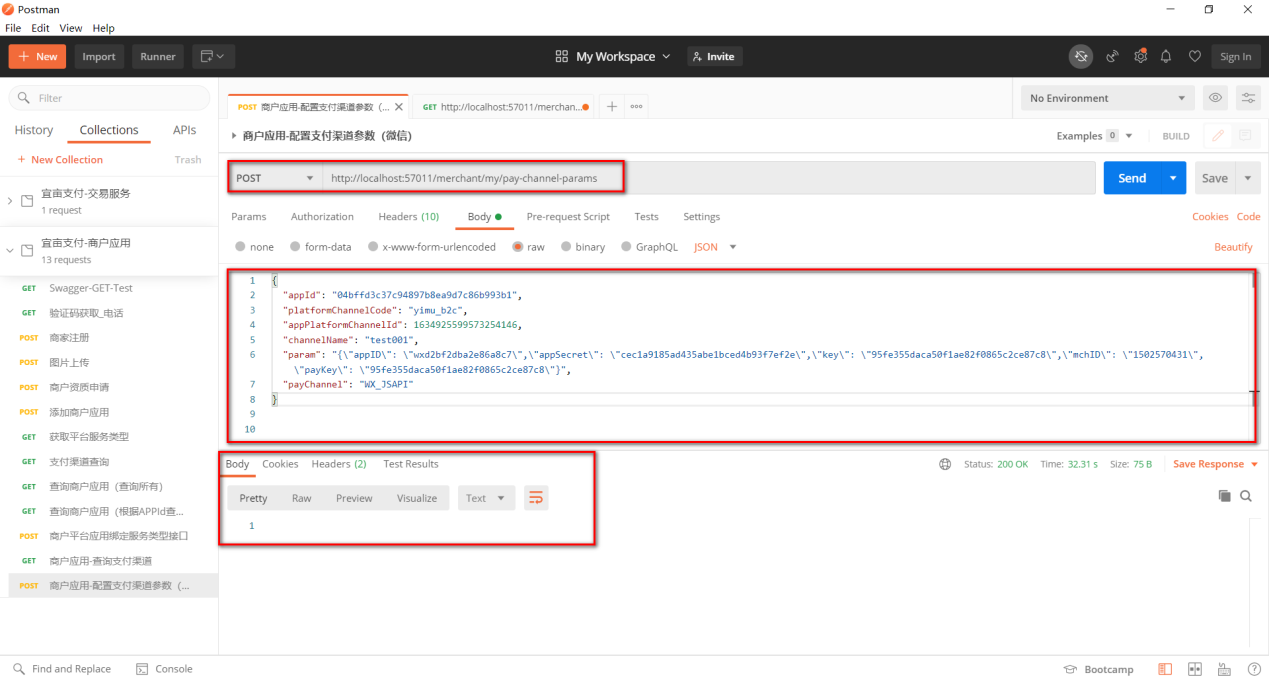
###### 3.5.5.2.1.微信

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" |

Body:

|  |
| --- |
| {    "appId": "04bffd3c37c94897b8ea9d7c86b993b1",    "platformChannelCode": "yimu\_b2c",    "appPlatformChannelId": 1634925599573254146,    "channelName": "test001",    "param": "{\"appID\": \"wxd2bf2dba2e86a8c7\",\"appSecret\": \"cec1a9185ad435abe1bced4b93f7ef2e\",\"key\": \"95fe355daca50f1ae82f0865c2ce87c8\",\"mchID\": \"1502570431\",\"payKey\": \"95fe355daca50f1ae82f0865c2ce87c8\"}",    "payChannel": "WX\_JSAPI"  } |



###### 3.5.5.2.2.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

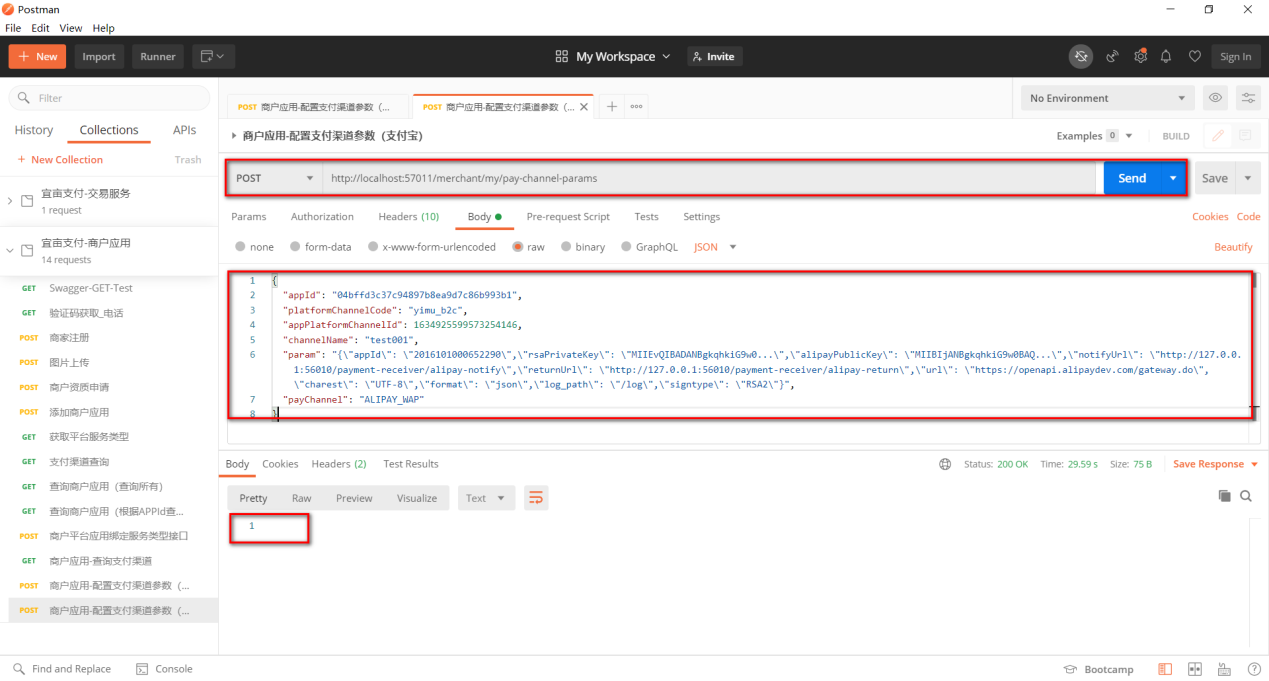
参照：[10.支付宝开放平台使用指南.docx](../00234002%20宜亩网（支付项目）2-各种手册/10.支付宝开放平台使用指南.docx)

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",  "charest": "UTF‐8",  "format": "json",  "log\_path": "/log",  "signtype": "RSA2"  }  "payChannel": "ALIPAY\_WAP" |

Body:

|  |
| --- |
| {    "appId": "04bffd3c37c94897b8ea9d7c86b993b1",    "platformChannelCode": "yimu\_b2c",    "appPlatformChannelId": 1634925599573254146,    "channelName": "test001",    "param": "{\"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\",\"charest\": \"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，并且用编辑器字符串双引号转义。

|  |
| --- |
| {\"appID\": \"wxd2bf2dba2e86a8c7\",\"appSecret\": \"cec1a9185ad435abe1bced4b93f7ef2e\",\"key\": \"95fe355daca50f1ae82f0865c2ce87c8\",\"mchID\": \"1502570431\",\"payKey\": \"95fe355daca50f1ae82f0865c2ce87c8\"} |

### 3.6.支付渠道参数查询

#### 3.6.1.交易服务渠道参数查询接口

##### 3.6.1.1.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; |

##### 3.6.1.2.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; |

##### 3.6.1.3.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) {  // 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));  **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**;  } |

#### 3.6.2.商户平台应用渠道参数查询接口

##### 3.6.2.1.接口定义1

接口描述：根据应用和服务类型获取原始支付参数列表

接口定义如下：

在PlatformParamController中定义queryPayChannelParam

|  |
| --- |
| @ApiOperation("根据应用和服务类型获取支付渠道参数列表")  @ApiImplicitParams({  @ApiImplicitParam(value = "应用id",name = "appId",dataType = "String",paramType = "path"),  @ApiImplicitParam(value = "服务类型代码",name = "platformChannel",dataType = "String",paramType = "path")  })  @GetMapping(value = "/my/pay-channel-params/apps/{appId}/platform-channels/{platformChannel}")  **public** List<PayChannelParamDTO> queryPayChannelParam(@PathVariable("appId")String appId,@PathVariable("platformChannel")String platformChannel){  **return** payChannelService.queryPayChannelParamByAppAndPlatform(appId,platformChannel);  } |

##### 3.6.2.2.接口定义2

接口描述：根据应用、服务类型及支付渠道获取原始支付参

接口定义如下：

在PlatformParamController中定义queryPayChannelParam

|  |
| --- |
| @ApiOperation("根据应用和服务类型和支付渠道获取单个支付渠道参数")  @ApiImplicitParams({  @ApiImplicitParam(value = "应用id",name = "appId",dataType = "String",paramType = "path"),  @ApiImplicitParam(value = "服务类型代码",name = "platformChannel",dataType = "String",paramType = "path"),  @ApiImplicitParam(value = "支付渠道代码",name = "payChannel",dataType = "String",paramType = "path")  })  @GetMapping(value = "/my/pay-channel-params/apps/{appId}/platform-channels/{platformChannel}/pay-channels/{payChannel}")  **public** PayChannelParamDTO queryPayChannelParam(@PathVariable("appId")String appId,@PathVariable("platformChannel")String platformChannel,@PathVariable("payChannel") String payChannel){  **return** payChannelService.queryParamByAppPlatformAndPayChannel(appId,platformChannel,payChannel);  } |

##### 3.6.2.3.接口测试

Postman

Get:

<http://localhost:57011/merchant/my/pay-channel-params/apps/04bffd3c37c94897b8ea9d7c86b993b1/platform-channels/yimu_b2c>

