## 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。

## 2.搭建redis环境

1、安装redis

参照：[2.Redis安装指南.docx](../00234002%20宜亩网（支付项目）2-各种手册/2.Redis安装指南.docx)

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: 3000ms |

在yimupay-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，yimupay-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yimupay-common工程

5.复制RedisCache到yimupay-transaction-service工程的com.me.yimu.pay.transaction.common.util包下。

6. 添加Redis配置文件，包：com.me.yimu.pay.transaction.config

|  |
| --- |
| **package** com.me.yimu.pay.transaction.config;  **import** org.springframework.context.annotation.Bean;  **import** org.springframework.context.annotation.Configuration;  **import** org.springframework.data.redis.core.StringRedisTemplate;  **import** com.me.yimu.pay.cache.Cache;  **import** com.me.yimu.pay.transaction.common.util.RedisCache;  /\*\*  \* redis配置类  \* **@author** PC  \*  \*/  @Configuration  **public** **class** RedisConfig {  @Bean  **public** Cache cache(StringRedisTemplate redisTemplate){  **return** **new** RedisCache(redisTemplate);  }    } |

## 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：

|  |
| --- |
| /\*\*  \* 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());**  **}** |

|  |
| --- |
| /\*\*  \* 根据应用和服务类型将查询到支付渠道参数配置列表写入redis  \* **@param** appId 应用id  \* **@param** platformChannelCode 服务类型code  \*/  **private** **void** updateCache(String appId,String platformChannelCode){  //得到redis中key(付渠道参数配置列表的key)  //格式：SJ\_PAY\_PARAM:应用id:服务类型code，例如：SJ\_PAY\_PARAM：ebcecedd-3032-49a6-9691-4770e66577af：shanju\_c2b  String redisKey = RedisUtil.*keyBuilder*(appId, platformChannelCode);  //根据key查询redis  Boolean exists = cache.exists(redisKey);  **if**(exists){  cache.del(redisKey);  }  //根据应用id和服务类型code查询支付渠道参数  //根据应用和服务类型找到它们绑定id  Long appPlatformChannelId = selectIdByAppPlatformChannel(appId, platformChannelCode);  **if**(appPlatformChannelId != **null**){  //应用和服务类型绑定id查询支付渠道参数记录  List<PayChannelParam> payChannelParams = payChannelParamMapper.selectList(**new** LambdaQueryWrapper<PayChannelParam>().eq(PayChannelParam::getAppPlatformChannelId, appPlatformChannelId));  List<PayChannelParamDTO> payChannelParamDTOS = PayChannelParamConvert.***INSTANCE***.listentity2listdto(payChannelParams);  //将payChannelParamDTOS转成json串存入redis  cache.set(redisKey, JSON.*toJSON*(payChannelParamDTOS).toString());  }  } |

## 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));  **return** PayChannelParamConvert.***INSTANCE***.listentity2listdto(payChannelParams);  } |