

RestFul 原则要求是：POST,GET,PUT 返回操作成功（新建的 || 获得的 || 修改状态后）的对象，delete 返回成功失败

logService 模块：

1、提供一个接口给各模块，管理员操作各模块后，调用写入日志的接口

POST /log

参数：@RequestBody Log log

Return : Log log

=====

goodsService 模块：

1、

给分享模块，判断商品是否在售

@GetMapping( "/goods/{id}/isOnSale" )

参数 @PathVariable Integer goodsId

Return Boolean

2、/\*12月18日下午15点修改\*/

根据 orderItemList 来修改库存。operation: true 代表加库存，false 代表减库存

@PUTMapping( "/product/list/deduct" )

参数 : @RequestBody List<OrderItem> orderItemList, @RequestParam

## boolean operation

Return :Boolean

### 3、/\*12月17晚21点修改\*/

作用： cartItem 里需要封装 Product, 但外部 api /product/{id} 只有管理员可用。

@GetMapping( "/user/product/{id}" )

参数： @PathVariable Integer productId

返回值： Product product

### 4、/\*12月19日下午16点新增\*/

作用： 提供一个接口给 collect、comment、footprint 调用，获取 goods 信息。（因为外部的/goods/{id}每次调用需要新增足迹，但是其他模块内部调用时不需要新增足迹，所以新增一个内部接口）

GET /inner/goods/{id}

参数： @PathVariable Integer id

返回值： GoosPo goodsPo

=====

discountService 模块：

### 1、/\*12.16 统一了接口进行复用\*/

Post /discount/orders

参数： Order order

return: Order order

paymentService 模块:

`/* 2019.12.17 修正 */`

这是一个昨天的疏漏 orz

- 1、说到由于 payment 模块增加了 payPayment() 方法，它使用 put /payment/{id} 的 url，与早先 updatePayment 的 url 重复，所以将 updatePayment 的 url 改做 put /payment/{id}/status
- 2、然后 getPaymentByOrderId 返回值应该是一个 List，因为预售可能一个 orderId 会产生多个 Payment，应该返回一个 List<Payment>

`/* 2019.12.15 更改 */`

- 1、删除 wxPaymentService 的 requestWxPayment 方法的 payChannel 参数,同时删除 paymentService 模块的 updatePayment 方法的 payChannel 属性

- 2、删除 paymentService 模块的 deletePayment(), getPayment()和 getAllPayments()方法。

`/* 2019.12.16 更改 */`

- 1、wxPayment 模块增加 refund 函数用于退款
- 2、updatePayment 方法新增 operationType 参数，区分支付和退款

`/* 2019.12.16 19:57 updating */`

`/* sorry for updating so frequently :( but according to our afternoon discussion with API criteria group, some changes have to be made. */`

- 1、为了支持预售功能（预售当前的逻辑是预售下单后，生成两个未支付的 payment 记录，并引导用户对其中一个 payment（即预付款）进行付款），将原先的 createPayment 方法拆分为 createPayment 方法与 payPayment 方法。注意的是，退款并不拆分，退款的代码仍然完整地保

留在 createPayment 方法中。

// 生成未支付 payment or 退款

@PostMapping("payment")

public Object createPayment(@RequestBody Payment payment)

返回 ResponseUtil 包装的 Payment 对象

// 对未支付 payment 进行支付

@PutMapping(payment/{id})

public Object payPayment(@PathVariable("id") Integer paymentId)

返回 ResponseUtil 包装的 Payment 对象

2、同时，order 模块支付为付款 payment 时，需要给 order 模块提供一个通

过 orderId 查找 payment 记录的接口。所以新增如下 url

// 根据 orderId 查找 payment

@GetMapping("payment/{id}")

public Object getPaymentByOrderId(@PathVariable("id") Integer  
orderId)

返回 ResponseUtil 包装的 List<Payment> 对象

wxPaymentService 模块：

- \* 支付模块调用此方法模拟微信统一支付 api
- \* 此方法应该返回给支付模块 prepay\_id 等信息

1、POST /wxpayment

参数      @RequestBody Payment payment

Return    String (paySn)

2、

```
/**
 * 用户发起最终支付
 * 此方法将调用 Payment 模块的 updatePayment 方法修改支付状态等信息
 * updatePayment 方法进而调用 Order 模块的 updateOrder 方法修改订单状态
 * 等信息
 * @param paySn
 * @Param endTime
 * @return WxPayment (只有两个字段 String paySn 和 Payment payment)
 */
@PutMapping("wxpayment/{id}/{status}")
public Object requestWxPayment(@PathVariable("id") String prepay_id,
    LocalDateTime endTime) ;
```

3、

```
/**
 * 退款
 * 当传入 payment 模块 addPayment 方法的参数 payment 的 actualPrice 属性
 * 为负时，addPayment 代表退款操作
 * 此时，addPayment 方法调用 wxPayment 模块的 refund 方法，传入 paySn 和
 * actualPrice (退款金额) 执行退款操作
 * 该方法与 requestWxPayment 类似，应调用 updatePayment 方法，进而调用
 * updateOrder 方法，修改相应的 (多个) 表状态
 *
 * @param refundWhom 支付时的 paySn
 * @param refundPaymentPaySn 退款时的 paySn
 * @param actualPrice 退款金额
 * @return
 */
@PutMapping("wxpayment/{id}/refund")
public Object refund(@PathVariable("id") String refundWhom, String
    refundPaymentPaySn, BigDecimal actualPrice)
```

paymentService 模块:

- \* （模拟的）微信后台调用此方法修改订单状态
- \* 此方法还会调用 order 模块的 updateOrder 方法，修改订单状态等信息
- \* 根据 operationType 确认此时是支付修改还是退款修改
- \* requestWxPayment 中调用时赋值“pay”
- \* refund 中调用时赋值“refund”

@PutMapping( “/payment/{id}” )

参数 @PathVariable String paySn, boolean successfulPayment, String

operationType

Return Payment

orderService 模块

1、提供接口给 AfterSale 查看 orderItem 是什么类型

@GetMapping ( “ordeltem/{orderId}/goodsType” )

参数 @PathVariable Integer orderId

Return goodsType (Integer)

/\*12.17 修改\*/

2、每天晚上搜索前一天 24 小时之内的完成的 grouponpo，然后对这些 po 都调用这个接口，根据 grouponpo 里的团购开始时间、结束时间和 goodsid 检查所有买了这个商品的 order 的数量，返给 discount 模块就行

GET ( “/orders/grouponOrders” )

参数 GrouponRulePo grouponRulePo

Return num

3、提供给 discount 模块用于团购退款。

/POST ( "/order/grouponOrders/refund" )

参数 @RequestBody GrouponRulePo grouponRulePo @RequestParam

BigDecimal price

Return boolean

4、还有一个内部接口，用于管理员设置预售失效后，要给订单退款。

这个是用在：预售活动进行到一半，管理员要设置活动失效。就要给已付款的定金退款。

由 discount 模块 告诉订单模块退多少钱，订单模块调用 payment 模块退钱并 修改订单状态等等

/POST ( "/order/presaleRule/refund" )

参数 PresaleRulePo presaleRulePo

Return boolean

5、给 payment 的回调接口(后面的状态表示付款还是退款)

PUT ("/order/{id}/paymentStatus)

参数 @PathVariable Integer id @RequestParam Integer state

return boolean

cartItemService 模块

1、提供给订单模块，通过 id 查相应 cart

@GetMapping( "/cartItem/{userId}" )

参数 @PathVariable Integer userId

Return     ArrayList<cartItem>

shareService 模块:

- 1、 提供一个接口给 Order 模块，每晚 12 点 order 模块给 share 模块 order 对象，让他计算该订单是否产生返点，share 模块计算后返回该返的返点数，供订单模块以后修改 user 模块返点

@GetMapping( "/rebate" )

参数       @RequestBody

Order order

Return     rebate(Integer)

userInfoService 模块

- 1、 修改返点，给订单模块调用

@PutMapping( "/user/rebate" )

参数@RequestParm Integer rebate

Return Integer rebate                   //返回修改结果，若结果为 0 是失败

- 2、 查看用户是否合法，提供给 share 模块调用

@GetMapping( "/user/validate" )

参数@RequestParam Integer userId

Return boolean beValid     //是否合法用户

freightService 模块



- 1、提供接口给 orderService 模块，orderService 模块给一个 Order，里面有 address，可以计算运费

/\*改为 post 请求\*/

@PostMapping( "/freightPrice")

参数      @RequestBody

Order order

Return    freightPrice (BigDecimal 类型)

logisticsService 模块

随便返回一个物流号

@GetMapping( "/logistics" )

参数无

Return    String 类型

/\*12.15 更新，删除参数 userId\*/

footprintService 模块

- 1、提供接口给 Goods 模块，用于增加足迹

@PostMapping("/footprints")

参数@RequestBody FootprintItemPo footprintItemPo

Return 产生的

FootprintItemPo footprintItemPo

