接口测试用例设计实践总结

by:授客 QQ：1033553122

设计思路

1） 优先级--针对所有接口

1、暴露在外面的接口，因为通常该接口会给第三方调用；
2、供系统内部调用的核心功能接口；
3、供系统内部调用非核心功能接口；

2） 优先级--针对单个接口

1、正向用例优先测试,逆向用例次之(通常情况，非绝对)；
2、是否满足前提条件 > 是否携带默认参值参数 > 参数是否必填 > 参数之间是否存在关联 > 参数数据类型限制 > 参数数据类型自身的数据范围值限制

3） 设计分析

通常，设计接口测试用例需要考虑以下几个方面：
1、是否满足前提条件
有些接口需要满足前置条件，才可成功获取数据。常见的，需要登陆 Token。
逆向用例：
针对是否满足前置条件(假设为 n 个条件)，设计 0~n 条用例

2、是否携带默认值参数
正向用例：
带默认值的参数都不填写、不传参，必填参数都填写正确且存在的“常规”值，其它不填写，设计 1 条用例；

3、业务规则、功能需求
这里根据实际情况，结合接口参数说明，可能需要设计 n 条正向用例和逆向用例

5、参数是否必填
逆向用例：
针对每个必填参数，都设计 1 条参数值为空的逆向用例

4、参数之间是否存在关联
有些参数彼此之间存在相互制约的关系
逆向用例：
根据实际情况，可能需要设计 0~n 条用例

5、参数数据类型限制
逆向用例：
针对每个参数都设计 1 条参数值类型不符的逆向用例

6、参数数据类型自身的数据范围值限制

正向用例：
针对所有参数，设计 1 条每个参数的参数值在数据范围内为最大值的正向用例

逆向用例：
针对每个参数(假设 n 个)，设计 n 条每个参数的参数值都超出数据范围最大值的逆向用例
针对每个参数(假设 n 个)，设计 n 条每个参数的参数值都小于数据范围最小值的逆向用例

以上几个方面考虑全的话，基本可以做到如下几个方面的覆盖：
主流程测试用例：正常的主流程功能校验；
分支流测试用例：正常的分支流功能校验。
异常流测试用例：异常容错校验

4） 编写描述
尽量逻辑化，这样方便后续的维护

5） 实践操作
接口样例

获取订单列表接口（多条件）

获取店铺指定期间的所有订单列表(多种条件组合)，默认根据日期倒序排序。

接口方向

客户端 -> 服务端

接口协议

接口地址：$xxx_Home/xxx/鉴权前缀/xxxxx/getAllOrderList

接口协议：JSON

HTTP 请求方式：GET

消息请求

字段列表如下：

字段名	数据类型	默认值	必填项	备注
shopId	int		是	商铺编号
token	string		条件	设备令牌。Token 鉴权方式必填
dateType	int	1	否	订单查询时间字段。 1：下单时间（order_time） 2：订单完成时间 （order_finish_time） 3：结算时间（shop_settle_time）
startDate	date		是	查询日期
endDate	Date		否	查询结束日期。
orderStatus	String		否	订单状态。 不填表示所有状态

				多个状态之间以英文逗号分割 0:已预定 1:已开单 2:派送中 3:已完成（原已结帐） 4:退单中 5:已退单 8:自助下单 9:待确认
orderTransactionType	Int		否	订单交易状态。 不填表示所有。 1:未完成， 2:已完成(3:已完成， 5:已退单)
payType	int		否	支付方式。 不填表示所有。 1:现金 2:POS 3:线上
cashierId	int		否	收银员
billerId	int		否	导购员
pNo	int		否	页码,从第1页开始，默认为1
pSize	int		否	每页记录数，默认为10

消息请求样例：

```
?shopId=1111111111&token=123411nmk515155&queryDate=2015-10-10
```

消息响应

字段元素如下：

字段名	数据类型	默认值	必填项	备注
orderTotalPriceTotal	double		是	实收金额合计（已完成的合计）
platformTotalIncomePriceTotal	double		是	平台服务费合计
cashPayTotal	double		否	现金支付（已完成的合计）
posPayTotal	double		否	POS 支付（已完成的合计）
onLinePayTotal	double		否	线上支付（已完成的合计）
lst	object		是	明细列表

明细列表对象字段元素定义：

字段名	数据类型	默认值	必填项	备注
orderId	string		是	订单 ID
orderTitle	string		是	订单标题

mobile	string		否	会员账号,如果是会员则显示手机号，为空时表示“非会员”
settlePrice	double		是	交易金额
orderTime	datetime		是	下单时间
serviceAmount	double		是	平台服务费
Status	Int		是	订单状态。 0:已预定 1:已开单 2:派送中 3:已完成（原已结帐） 4:退单中 5:已退单 8:自助下单 9:待确认
cashPay	double		否	现金支付
posPay	double		否	POS 支付
onLinePay	double		否	线上支付

成功时，返回 JSON 数据包：


```
{
    "code": 0,
    "msg": "查询订单列表成功！",
    "data": {
        "pNo": 1,
        "rCount": 5,
        "orderTotalPriceTotal": 23.3,
        "platformTotalIncomePriceTotal": 0,
        "lst": [
            {
                "orderTitle": "kouxiangtang",
                "settlePrice": 15.89,
                "cashTotal": 15.89,
                "posTotal": 0,
                "onLineTotal": 0,
                "orderTime": "2015-09-29 13:44:26",
                "orderId": "12345679282015092913440268141",
                "mobile": "13424183952"
            },
            {
                "orderTitle": "红塔山",
                "settlePrice": 7.5,
                "cashTotal": 7.5,
                "posTotal": 0,
```

```
                "onLineTotal": 0,
                "orderTime": "2015-09-29 11:37:58",
                "orderId": "123456792820150929113705882 73"
            }
        ]
    }
}
```

用例设计

1	获取订单列表接口	getAllOrderList	test-N-所有默认值参数都不填	shopId 参数值：正确且存在； startDate 参数值：正确
2			test-N-带设备token查询	shopId 参数值：正确且存在； token 参数值：正确且存在； startDate 参数值：正确
3			test-N-按订单时间类型查询-下单时间	shopId 参数值：正确且存在； startDate 参数值：正确 dateType 参数值：1
4			test-N-按订单时间类型查询-完成时间	shopId 参数值：正确且存在； startDate 参数值：正确 dateType 参数值：2
5			test-N-按订单时间类型查询-结算时间	shopId 参数值：正确且存在； startDate 参数值：正确 dateType 参数值：3
6			test-N-按查询结束日期查询-单天	shopId 参数值：正确且存在； startDate 参数值：正确 endDate 参数值：与startDate相同
7			test-N-按查询结束日期查询-跨天	shopId 参数值：正确且存在； startDate 参数值：正确 endDate 参数值：与startDate不同
8			test-N-按订单状态查询-单个状态	shopId 参数值：正确且存在； startDate 参数值：正确 orderStatus 参数值：多个状态
9			test-N-按订单状态查询-多个状态	shopId 参数值：正确且存在； startDate 参数值：正确 orderStatus 参数值：多个状态
10			test-N-按交易状态查询	shopId 参数值：正确且存在； startDate 参数值：正确 orderTransactionType 参数值：单个状态
11			test-N-按支付方式查询	shopId 参数值：正确且存在； startDate 参数值：正确 payType 参数值：单种支付方式
12			test-N-按收银员查询-收银员存在	shopId 参数值：正确且存在； startDate 参数值：正确 cashierId 参数值：正确且存在的收银员
13			test-N-按收银员查询-收银员不存在	shopId 参数值：正确且存在； startDate 参数值：正确 cashierId 参数值：正确，但不存在的收银员

14			test-N-按导购员查询-导购员存在	shopId 参数值：正确且存在； startDate 参数值：正确 cashierId 参数值：正确且存在的导购员
15			test-N-按导购员查询-导购员不存在	shopId 参数值：正确且存在； startDate 参数值：正确 cashierId 参数值：正确，但存在的导购员
16			test-N-按页码查询-存在页码	shopId 参数值：正确且存在； startDate 参数值：正确 pNo 参数值：正确且存在的页码
17			test-N-设置页面容量	shopId 参数值：正确且存在； startDate 参数值：正确 pSize 参数值：正确
18			test-N-条件组合	所有参数 参数值：正确，符合规则
19			test-E-按商铺id查询-商铺id不存在	shopId 参数值：正确，但不存在； startDate 参数值：正确
20			test-E-带设备token查询-token不存在	shopId 参数值：正确且存在； token 参数值：正确，但不存在； startDate 参数值：正确
21			test-E-按商铺id查询-商铺id为空	startDate 参数值：正确
22			test-E-按查询起始日期查询-日期为空	shopId 参数值：正确且存在
23			test-E-按时间类型查询-时间类型不在定义范围内	shopId 参数值：正确且存在； startDate 参数值：正确 dateType 参数值：不在定义范围内的合法值
24			test-E-按订单状态查询-订单状态不在定义范围内	shopId 参数值：正确且存在； startDate 参数值：正确 orderStatus 参数值：不在定义范围内的合法值
25			test-E-按订单交易状态查询-交易类型不在定义范围内	shopId 参数值：正确且存在； startDate 参数值：正确 orderTransactionType 参数值：不在定义范围内的合法值
26			test-E-按支付方式查询-payType不在定义范围内	shopId 参数值：正确且存在； startDate 参数值：正确 payType 参数值：不在定义范围内的合法值
27			test-E-查询结束日期小于查询起始时间	shopId 参数值：正确且存在； startDate 参数值：正确 endDate 参数值：小于startDate

28			test-E-按页码查询-页码不存在	shopId 参数值：正确且存在； startDate 参数值：正确 pSize 参数值：正确 pNo 参数值：正确，但不存在的页码
29			test-E-按商铺id查询-商铺id非int型	shopId 参数值：非int型 startDate 参数值：正确
30			test-E-按设备token查询-token非string类型	shopId 参数值：正确且存在 startDate 参数值：正确 token 参数值：非string类型
31			test-E-按订单时间类型查询-时间类型非int型	shopId 参数值：正确且存在； startDate 参数值：正确 dateType 参数值：非int值
32			test-E-按起始日期查询-时间类型非date型	shopId 参数值：正确且存在； startDate 参数值：非date endDate 参数值： 正确
33			test-E-按结束日期查询-时间类型非date型	shopId 参数值：正确且存在； startDate 参数值：正确 endDate 参数值： 非date
34			test-E-按订单状态查询-订单状态非string类型	shopId 参数值：正确且存在； startDate 参数值：正确 orderStatus 参数值：非string类型
35			test-E-按交易状态查询-交易状态非int型	shopId 参数值：正确且存在； startDate 参数值：正确 orderTransactionType 参数值：非int值
36			test-E-按支付方式查询-支付方式非int值	shopId 参数值：正确且存在； startDate 参数值：正确 payType 参数值：非int值
37			test-E-按收银员查询-收银员id非int值	shopId 参数值：正确且存在； startDate 参数值：正确 cashierId 参数值：非int值
38			test-E-按导购员查询-导购员id非int值	shopId 参数值：正确且存在； startDate 参数值：正确 cashierId 参数值：非int值
39			test-E-按页码查询-页码非int值	shopId 参数值：正确且存在； startDate 参数值：正确 pNo 参数值：非int值
40			test-E-按商铺id查询-商铺id为空	startDate 参数值：正确
41			test-N-按参数类型最大值查询	所有参数 参数值：参数值存在，且为参数类型最大值，比如32位 int最大值：2147483647
42			test-E-按商铺id查询-商铺id超过类型范围值	shopId 参数值：值超过int类型的最大值 startDate 参数值：正确

43			test-E-按订单状态查询-订单状态值超过类型最大值	shopId 参数值：正确且存在； startDate 参数值：正确 orderStatus 参数值：值超过string类型的最大值
44			test-E-按交易状态查询-交易状态值超过int类型最大值	shopId 参数值：正确且存在； startDate 参数值：正确 orderTransactionType 参数值：非int值
45			略	

存在问题：

如上，还没写完就有 40 几条用例了，要是接口参数再多点，接口数量再增加点，工作量可想而知，所以，问题来了，咋办呢？

个人见解：

1、根据接口的使用对象(外部，系统内部)，有选择的去、留部分用例

2、根据接口的是否核心接口，有选择的去、留部分用例

3、根据参数说明，及实际情况，有选择的去、留部分用例

实例：

上例这个接口，是供 app、商铺后台调用的，且为系统内部调用，所以，以下用例可酌情略去：

test-E-按商铺 id 查询-商铺 id 非 int 型

test-E-按设备 token 查询-token 非 string 类型
test-E-按订单时间类型查询-时间类型非 int 型
test-E-按起始日期查询-时间类型非 date 型
test-E-按结束日期查询-时间类型非 date 型
test-E-按订单状态查询-订单状态非 string 类型
test-E-按交易状态查询-交易状态非 int 型
test-E-按支付方式查询-支付方式非 int 值
test-E-按收银员查询-收银员 id 非 int 值
test-E-按导购员查询-导购员 id 非 int 值
test-E-按页码查询-页码非 int 值

理由：
这个接口是给其它开发于系统内部调用的，开发过程中，开发者肯定需要调用这些接口，如果类型错了，他们也就获取不到预期的数据，这些错误，他们肯定可以发现，所以，他们传递的参数值一般能保证类型正确。

test-N-按参数类型最大值查询所有参数
test-E-按商铺 id 查询-商铺 id 超过类型范围值
test-E-按订单状态查询-订单状态值超过类型最大值
test-E-按交易状态查询-交易状态值超过 int 类型最大值
略去的用例部分（参数值超过类型最大值）

理由：
1、内部调用，参数值不是外部手动输入的，输入数据长度、值大小可控，当然如果数据一直增长，那再大的类型可能都无法保证不超出，比如自动增长的商铺 id
2、部分参数的参数值是自定义的，比如 订单时间类型，就那几种，除非传错了，不然不可能超出范围

最后简化后的用例数差不多 28 条，如果是手工测试，对于正向用例，根据等价类原理，可以制造一条数据，覆盖多条用例，当然，也可以冗余处理，即一条用例一条数据，这样的好处就是每次的验证点比较单一一点，比较有针对性。

问题
如果是自动化测试呢，这里是设计一个方法覆盖多条用例呢(如上，一条数据，覆盖多条用例)？还是一个方法覆盖一条用例呢？

我个人的答案是一个方法一条用例，你的呢？