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": "12345679282015092911370588273"

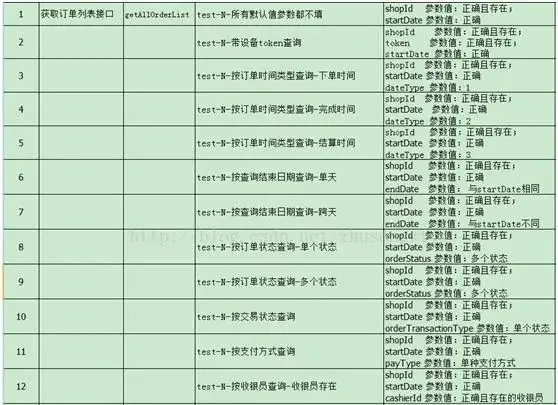
            }

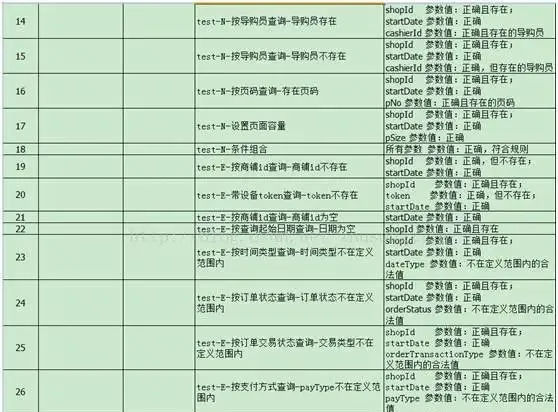
        ]

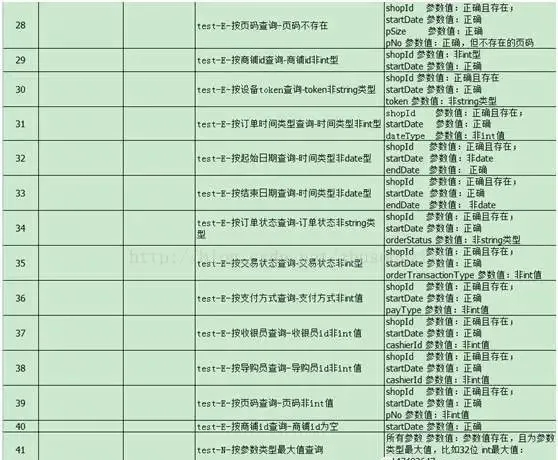
    }

}

用例设计







IMG_256

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？