

2.1.1 数值型

数值型的参数主要考虑以下几个方面设计：

如果参数规定了值的范围，则需要考虑等价类取值范围内、取值范围外，取值的边界，如有需要，可能会遍历取值范围内的各个值。

例如检查权限的接口：TaskChecker.checkTask(int taskID) taskID的取值范围是1-35，那么设计时考虑：

1-35范围内和范围外的值；

1-35的边界：0，1,35,36；

类型的特殊值：-1,0

数据类型的边界值：int的最小值最大值；

因为1-35代码的权限ID不同，可能需要遍历1-35的每个值。

常见问题和风险：

特殊值处理不当导致程序异常退出；

类型边界溢出

取值范围外值未返回正确的错误信息等

2.1.2 字符串型

字符串型的参数，主要考虑字符串的长度和内容：

例如接口转换设置闹钟的接口DateUtil.getDayOfDDHH(String ddhh)，用例可以考虑：

长度为4位，比4位少，比4位多；

边界值：String的最大长度；

特殊值：空字符；

字符串内容可考虑类型：数字，非数字；

特殊字符。

如果是输入用户输入且其他用户可见的内容，则还需要考虑敏感字是否被正常过滤。

可能出现的问题和风险：

传入非特定类型程序异常退出

超长字符未进行处理，导致存储、显示等异常

其他用户可见设置的敏感字

2.1.3 数组或链表类型

参数类型为数组或链表时，用例可以考虑：

例如批量提交任务的接口submitTask(int[] taskID)，参数用例设计考虑：

正常取值：1-5个权限，范围外：6个权限；

边界值：1-35的边界值，请求允许最大最小值；

特殊值：0个；

合法ID和不合法的；

重复的ID等。

可能存在的问题和风险：

0个item时程序异常退出；

重复的item处理时未去重导致结果异常等。

2.2 针对逻辑设计

接口需要进行一些逻辑处理的，那么按逻辑设计用例可以从以下几个角度分析。

2.2.1 约束条件分析

（1）数值限制：分数限制、金币限制、等级限制等等。

例如：兑换Q币活动要求积分>50才可参与。

（2）状态限制：登录状态等。

例如：同步用户信息需要先登录账号。

（3）关系限制：绑定的关系，好友关系等。

例如：帮家人防骗功能只能查询绑定家人的来电信息。

（4）权限限制：管理员等。

约束条件的测试在功能测试中经常遇到，在接口测试中更为重要。它的意义在于：用户进行操作时，在该操作的前端可以已经进行了约束条件的限制，故用户无法直接触发请求该接口。但是实际上，如果有其他手段：例如UI有bug或者通过技术手段直接调用接口，那么接口是否针对这些条件进行了限制就尤为重要。

例如常见的例子：要兑换5Q币需要200积分，但是我积分不足，所以兑换按钮是灰色无法点击的状态：

2.2.3 状态转换分析

被测逻辑可以抽象成状态机，各个状态之间根据功能逻辑从一个状态切换到另一个状态。如果我们打乱了这个次序，从一个状态切换到另一个不在它下一状态集中的状态，那么逻辑将会打乱，就会出现逻辑问题。

如上图所示，从某状态改变到新的状态，依赖于转换接口。而对于某转换接口，其输入状态是确定的，比如Fun23, 这个函数只能把状态2转换为状态3，而不能把状态1转换为状态3。那么测试点就可以是：

（1）状态为状态2，调用接口Fun23（），状态切换到状态23；

（2） 状态为1,3等，调用接口Fun23（），状态不能切换。

例如在做任务的时候，任务有三种状态：未领取，已领取未提交，已完成三种状态。

那么可以这样设计：

（1）正常的状态切换：未领取状态，领取任务后变为已领取状态；已领取满足任务条件提交后，变成已完成状态；完成后可以再次领取任务。

（2） 非正常的状态切换：未领取任务满足任务条件直接提交任务；已领取时再次领取任务等等。

常见的问题和风险：

可通过特殊手段达到原本不能的状态，从而谋取利益。

**2.3 针对输出设计**

针对输出设计其实是针对接口返回的结果进行分析。

2.3.1 针对输出结果

接口处理正确的结果可能只有一个，但是错误异常返回结果有很多情况很多值。如果知道返回结果有很多种，就可以针对不同结果设计用例。例如提交积分任务的时候我们通常能想到的是返回正确和错误，错误可能想到：无效任务，无效登录态，但是不一定能否完全覆盖所有错误码，而接口返回定义的返回码可以设计更多用例：

覆盖返回码也是用例设计的一种思路。

常见问题和风险：

（1）错误前端处理不足，导致前端异常；

（2）错误提示处理不当，导致用户看到晦涩的错误码；

（3）错误提示不当，导致用户不知道哪里出了问题，如何解决。

**2.3.2 接口超时**

接口正常情况下是有返回的，那么如果接口不返回呢？也就是说接口超时后的处理也是测试需要考虑的部分。如果超时处理不当，可能会引起以下问题：

（1）未进行超时处理，导致整个流程阻塞

（2）超时后又收到接口返回，导致逻辑出现错乱

2.4 其他测试设计

2.4.1 已废弃接口测试

已废弃协议，是指之前有定义，但是因为需求变更或其他原因，目前版本不用。这些接口虽然不再使用，但有可能代码并没有及时删除。如果利用技术手段调用这些接口，可能获取额外利益。

例如，任务之前有个清理任务，在一个版本需求里将清理任务替换为下载任务。在新版本客户端已不再调用完成清理任务的接口；但是如果该接口未关闭，用户就可以继续请求submitTask(int taskID)接口完成清理任务获得积分。

因此新版本在考虑兼容旧版本的同时，还应做好相关废弃接口的检查，避免用户获得额外利益。

**2.4.2 接口设计合理性分析**

接口定义是否合理可以从以下几个方面分析：

（1）接口字段是否冗余；

（2）接口是否冗余；

（3）接口是否返回了调用方期望得到的信息；

（4）接口定义是否可满足所有调用需求；

（5）接口定义调用是否方便。

用例设计：

第一个方向：

* 用例一
  + 基本功能，全部参数

1.参照接口文档，了解接口功能后，首先判断是否符合业务需求逻辑  
2.验证基本功能，将涉及到的参数全部传参

* 用例二
  + 缺少可选项

1.传参中只有必填项，无可选项参数  
2.传参中只有必填项，缺少其中一个可选项参数

* + 缺少必填项

1.传参中缺少其中一个必填值，参照变1原则。  
2.传参中缺少全部必填参数，有可选项  
3.将必填参数名称修改为非法

* 用例三
  + 参数越界

1.探测参数的边界值，通过边界值检测接口

* 用例四
  + 参数类型

1.字符串、整型

* 用例五
  + 覆盖全部路径，包括正常、异常
* 用例六
* 入库出库检查

入库a，库中值为：a，查询值为a

引言：

不知不觉中成为一名测试攻城狮已将近7年之久，在这期间做过不少API接口测试。 最早接触的API接口在项目中称之为（webservice)，期间走了不少弯路。也收获了不少的经验，现分享下在测试的路上走过的坑。

第一阶段小白初体验:

初始接触API是在11年，第一次接触API接口一脸的懵逼，很茫然不知如何下手测试，对http网络协议也不了解。后来 由研发同学提供对应接口 web界面，测试时进入对应接口界面，在textbox输入对应参数值 提交后查看返回值。但刚幵始一脸懵逼，返回值也不清楚是否正确，每次都是把研发同学叫 过来直接看结果。严重耽误了效率,后来不断的自我反思，在研发同学的不断讲解API知识。 不断前行最终自己可以独立测试，虽然中间过程也比较坑。比较囧(在这里特别感谢研发 的同学非常耐心的讲解API知识，也非常感谢大芽同学的帮助 ）

第二阶段菜鸟阶段

经过不断的摸索和不断学习API知识，提高自己。从http网络协议入手，到一次网络请求的工作过程。如下图:

HTTP网络协议，目前常用的请求类型：POST；GET；PUT等。目前用的比较多的是POST,GET这2种。一般涉及到写操作使用POST请求，只是读操作使用GET请求。

通过不断测试API接口和阅读API接口开发文档，总结经验设计API功能测试用例。例如初始时设计发表评论接口： 设计Case如下 1.有Userid，有评论内容 2.无Userid，有评论内容 3.有Userid，无评论内容 4.无Userid，无评论内容 5.有Userid,有评论内容（查询数据库，是否存储一致）6.有Userid,评论内容等于限制数 7.有Userid,评论数大于限制数。

用例设计完成，剩下的就是执行用例。不断的总结经验，根据发现的Bug，补充自己的测试用例。在测试过程中发现了Bug,尽量描述全。把请求的参数，以及返回的参数全贴到Bug描述里面，这样研发在复现Bug，也很节省时间。这样也是提高了大家的工作效率。使用Excel比较不错，设计用例时候排列组合就好。

第三阶段工具运用：

在繁重的测试过程中发现这种由研发提供的web界面方式效率比较低下，回归测试时 非常不便捷。测试数据每次都要重新输入，增加了大量的重复工作。每次回归测试都需要消耗大量的时间，主要是测试数据的管理在WEB页面没办法保存。通过网上查找和同事咨询，引入了接口测试工具 SoapUI。在SoapUI中创建API的项目，添加对应API接口。创建API接口测试用例，把测试数据配置好就可以了。在执行测试任务时，执行对应的API接口测试用就可以了。使用SoapUI后重复输入的工作解放了出来。提高了非常不错的工作效率，节省了不 少工期.

第四阶段菜鸟业务提升

在测试过程发现自己设计的用例，数量上太过于庞大。而且大多数用例都是用来测试接口本身，反而忽略了接口本身的业务。

例如login接口，设计功能用例： 1.有帐号，有密码 2.有帐号，无密码 3.无帐号，无密码 4.无帐号，无密码 5.有帐号，有密码 （登录用户，与数数据库校验一致）

但实际的用户可不单单一类用户，按照注册来分可以分为新注册用户，老用户。按照权限来分比如京东，分为普通会员，铜牌会员、银牌会员、金牌会员、钻石会员。所以实际测试需要测试 这2大类用户登录。这个时候需要提炼出实际业务测试用例，实际的业务有的是不单调一个接口，是多个接口组成。中间还涉及到B接口需要从A接口获取参数。

设计业务用例：1.新注册会员登录 2.普通会员（老用户）登录 3.铜牌会员登录 4.银牌会员登录 5.金牌会员登录 6钻石会员登录 7.非注册用户登录。

确定了测试用例，剩下的就按照测试用例执行就可以了。