目 录 (五十五期·上)

测试女巫紧跟时代脉搏 AI 之三种聊天机器人

◆ 作者：王平平

一、前言

AI 是一个非常火的名词，不管是公司内部还是放眼看看外面的世界，真的有一种满世界 AI 的感觉，风来了，即使没办法站在风口上，像猪一样飞起来；也总要走出你的山洞感受一个这股风潮吧。更何况我们介绍了这么多有关 Python 的知识，对于 AI，Python 又是家人般的存在，我们怎能置身事外呢？其实最近女巫一直在关注这些热点的内容：爬虫，AI，大数据分析；这些技术都与之前学习的内容都能连接起来，真的再火的新技术，再高大上的功能都不是凭空出现，例如大数据分析，我竟然看到了 6 sigma 的知识，python 中有关大数据分析的各个模组，竟然可以与 Minitab 软件的各个功能是可以达到某种程度的一一对应。我真的很想尖叫哈哈，就是这样毫无预期地转角遇到爱，怎么不让人兴奋，所以不好意思，everybody 后续的几期我想总结这些热点内容，对于系统架构的总结，我们可以需要把 schedule delay 几期哈~

我们做三个对比：对于聊天机器人，如果使用非 AI 技术实现与 AI 简单的技术实现，以及利用外部 AP 将如何实现。让大家对于 AI 技术与非 AI 技术有一个基本的认识，目前很多公司都会免费提供一些有关 AI 的 API 无偿给大家使用，我们利用这些 API 会效率很高地实现非常酷炫的功能。

好吧，不多说，我们开始吧~

二、聊天机器人简单介绍

1. 常见的聊天机器人们

Q&A 系统：IBM Waston

对话系统(Dialog System):Siri，Amanzon Alexa

聊天机器人 (chatbot): 微软小冰，小爱同学，小度

2. 搭建聊天机器人的四种方法

- 基于模板匹配的方法(非 AI 方法): 此次讲解
- 基于搜索的方法(AI 的较简单的方法) : 此次讲解的重点
- 基于意图识别的方法(AI 的较复杂的方法): 后续学习后再给大家讲解
- 生成式方法(i.e. ，端到端): 后续学习后再给大家讲解

三、基于模板匹配的方法(非 AI 方式实现聊天机器人的方式)

1. 方法说明

- 它的核心就是定义一条条的规则，算法
- 不需要懂 AI 技术即可以实现
- 最初的 ALICE 聊天机器人系统（1995 年）是最经典的案例

2. 代码范例

即使用 flask server 搭建模板驱动式的聊天机器人

通过连接中的<data>传进来的参数，来与一堆 if 和 elif 定义的模板做匹配，匹配到就返回模板中写死的一些回复

不用想都知道，如果想实现较为复杂的对话系统，下面会有多少 if 和 Elif

```
from flask import Flask
import requests
app = Flask(__name__)
# 装饰器中为访问的路径
# @app.route("/greedyai/<data>")
@app.route("/greedyai/<data>")
def hello_world(data):
    if "美丽" in data:
        return "白雪公主"
    elif "通讯" in data:
        return "哈哈通讯公司"
    elif "测试" in data or "开发系统" in data:
        return "系统部总有一款适合你"
    elif "北京团队" in data:
        return "系统二部"
    elif "南京团队" in data:
        return "系统四部"
    elif "框架" in data:
        return "AI系统哈哈哈哈
    else:
        "哎呦，小主我不懂你的意思哎~"
    return data
if __name__ =='__main__':
    app.run()
```

3.存在的问题

- 需要设计大量的规则，且要保证规则之间没有冲突(越复杂越困难)

- 可扩展性很差，重复工作很多

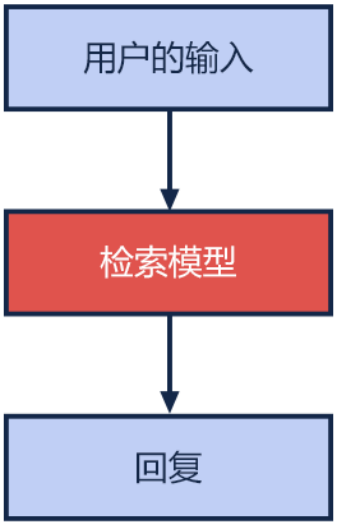
- 很难处理语句的多样化表达(过于死板)

如果你想问一些稍微复杂的问题，例如“不美丽的人是谁”，这个代码还是会回答:“白雪公主”，因为它无法处理这样的问题。

四、基于搜索的方法（自己实现的 AI 类型的 Chat Robot）

1. 原理

在输入和回复之间建立一个检索模型(数据公式)



2. 建立检索的模式

1) 计算两个句子之间的相似度

数学公式：余弦相似度 d = s1 · s2 /(|s1|*|s2|)

s1 · s2 是 s1 和 s2 的内积

|s1|*|s2|是 s1 和 s2 各自的长度(范数)相乘，结果是 0 到 1 之间，数值越大，相似度越高

如下图:

$$\frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

s1 = (x1,x2,x3...)

s2 = (y1,y2,y3...)

$$d = \frac{x1y1+x2y2+x3y3.....}{\sqrt{x1^2+x2^2+x3^2.....}\ \sqrt{y1^2+y2^2+y3^2.....}}$$

3. AI 编程思路

1) 人工智能(AI)是什么？

就是将自然界的问题转换为数学问题。对于“计算两个句子之间的相似度”如何转换为数学问题；就是 AI coding 的一个简单而经典的思路。

所谓转换数学问题就是如何将句子中的单词转换为数字，然后才能使用余弦相似度数据公式计算它们的相似度。

2) 词典

用数学的方式(list 数据的方式)来表达单词库

我们 启发 通讯 开发系统 开发团队 测试团队 南京 香港

[1 1 1 1 1 1 1 1]

基于上述 8 个单词，组成了 8 个数据的的数据表示法。

3) 单词 list 数据表示法

“启发”的 list 的数据表示法：[0，1，0，0，0，0，0，0]

“香港”的 list 的数据表示法：[0，0，0，0，0，0，0，1]

“南京”的 list 的数据表示法：[0，0，0，0，0，0，1，0]

4) 句子的 list 的数据表示法

我们 是 属于 启发 的 测试团队 [1，1，0，0，0，1，0，0]

方法：以词典为基准，一一比对，词典中和句子中都有的单词，数字为 1；否则为 0

4、AI coding 步骤

1）将句子拆分成多个分开的单词

模组：jieba(拆分句子的模组)

根据以下 study 的结果可以看出你可以根据自己的需要选取不同方式的拆分句子的方式；对于一些特殊拆分的方式，需要另外写算法，例如你不希望将“测试团队”拆分成两个单词，需要在拆分前，提前将“测试团队”提出来另外处理；

```
import jieba
s1 = "我们是属于启发的测试团队"
# 第一种切分模式：精确模式。比较适合做文本分析的
s1_result_list = list(jieba.cut(s1))
print("精确模式",s1_result_list)

# 全模式将有可能
s1_result_list = list(jieba.cut(s1,cut_all =True))
print("全模式",s1_result_list)

#搜索引擎模式
# 首先先匹配title，如果title没有，再匹配快照(快速解释)
s1_result_list = list(jieba.cut_for_search(s1))
print("搜索引擎模式",s1_result_list)
```

运行结果:

```
精确模式 ['我们', '是', '属于', '启发', '的', '测试', '团队']
全模式 ['我们', '是', '属于', '启', '发', '的', '测试', '团队']
搜索引擎模式 ['我们', '是', '属于', '启基', '的', '测试', '团队']
```

5.使用 numpy 将一维的 list 转成至少 2 维的 list

1) 来自灵魂的拷问

为什么要转成 2 维以及 2 维以上的 list?

因为机器学习的相关模块，也就是说数学公式(余弦相似度)，需要的 input，是二维起步，人家不识别一维的 list，所以在做机器学习或者有关 AI 的相关公式计算，必须先用 numpy 进行数据的转换。

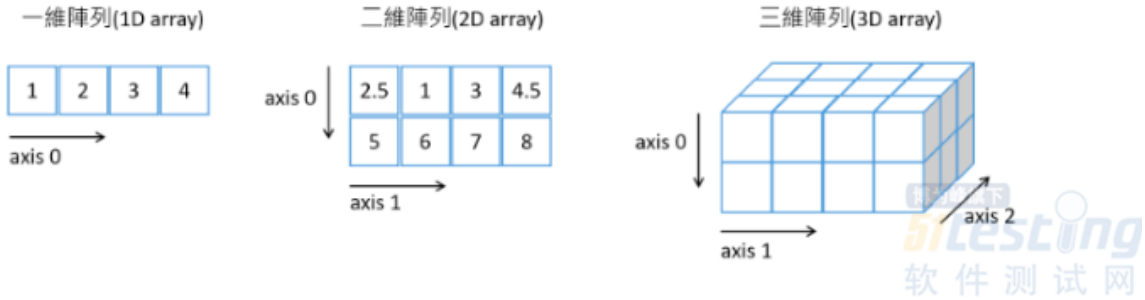
2) 维度的意义

一个点是一维；二维是一条线(线由无数点组成)，一个面是由无数条线组成一个实物；三维由无数面组成真实的世界；四维由无数三维物体组成(可以理解为目前我们真实的世界加上一个时间的空间这一维度；可以想象这个世界上有多个平行的世界存在，通过时间这个维度，可以在多个平行的世界随意穿梭：任意门的感觉，有木有= =)

3) numpy 的模组

在资料科学(Data science)和机器学习(Machine Learning)时，利用 Numpy 的阵列操作是非常重要的，因为 Data science 和 Machine Learning 的主要功能都构建在多重维度的 ndarray 之上

一到三维的阵列如下图:



a.建立阵列

```
In [4]: import numpy as np

In [5]: np.array([1,2,3])
Out[5]: array([1, 2, 3]) # 通过list到numpy.array()建立一维阵列

In [6]: np.array((1,2,3))
Out[6]: array([1, 2, 3]) # 通过tuple到numpy.array()建立一维阵列

In [7]: np.array([[4,5],[6,7]])    #建立二维阵列
Out[7]:
array([[4, 5],
       [6, 7]])
```

b.建立有初始值的阵列

```
np.zeros((2, 3))                # 建立一個2x3全為0的陣列
np.ones((2, 3, 4))              # 建立一個2x3x4全為1的陣列
np.arange(1, 10, 2)             # 建立一個由1開始，不超過10，間隔值為2的均勻數值陣列
np.linspace(0, 10, 5)           # 建立一個0到10之間，均勻的5個數值陣列
np.full((3,2), 8)               # 建立一個3x2全為8的陣列
np.eye(2)                       # 建立一個5x5的單位矩陣
np.random.random((2,3))         # 建立一個2x3的隨機值矩陣
```

c.对于已有的阵列进行维度重新定义

```
In [11]: a = np.array([1,3,5,7,9,0])  #一维的array
In [12]: a.reshape(3,2)     # 将一维变为三行两列
Out[12]:
array([[1, 3],
       [5, 7],
       [9, 0]])
In [13]: a.reshape(1,-1)  # 将一维变成二维，但是第二行是什么也没有，实际上只有一行
Out[13]: array([[1, 3, 5, 7, 9, 0]])
```

总结上述知识，可以产生如下代码

```
vector_list = []
word_vector_list = ["我们","启发","通讯","开发系统","开发团队","测试团队","南京","香港"]
import numpy as np
def get_vector(data):
    vector_list = []
    for i in word_vector_list:
        if i in list(jieba.cut(data)):
 # 将传进来的句子进行分割，并进行遍历，如果在我们定义的字典中则向量list为1，否则为0，就是句子翻译成数字的过程
            vector_list.append(1)
        else:
            vector_list.append(0)
    print('vector_list',vector_list)
    return np.array(vector_list).reshape(1,-1) # 二维，第二维是空
 # 将一维的list转成二维的array
```

将需要转换的句子，转成 vector

```
s1 = "我们是属于启发的测试团队"
s2 = "我们是启发的南京开发团队"
s3= "我是哈哈"
question = "你是启发的吗"

question_vector_list = get_vector(question)
s1_vector_list = get_vector(s1)
print("s1_vector_list",s1_vector_list)
s2_vector_list = get_vector(s2)
print("s2_vector_list",s2_vector_list)
s3_vector_list = get_vector(s3)
print("s3_vector_list",s3_vector_list)
```

6.计算文本相似度

学习的模组：sklearn

计算相似度的方法有很多种:

欧式距离; 余弦相似度; 曼哈顿距离; 切比雪夫距离; 闵可夫斯基距离; 马氏距离……

我们这次用的是余弦相似度。需要使用的函数名称是: cosine_similarity(a,b)

a 和 b 是经过 numpy 转成二维及其以上的 array。

```
from sklearn.metrics.pairwise import cosine_similarity

print(cosine_similarity(question_vector_list,s1_vector_list))
print(cosine_similarity(question_vector_list,s2_vector_list))
print(cosine_similarity(question_vector_list,s3_vector_list))
```

运行结果如下:

```
[[0.70710678]]
[[0.57735027]]
[[0.]]
```

数组越大相似度越高

7. AI 聊天机器人后端实现

1）需求描述

假想我们是一个做教育的公司，我们需要做一个简单得聊天机器人，来回答一些重复性比较高的问题。

2）使用技术描述

- flask 后端框架(最基础的即可)

- numpy 的 reshape(一维转二维); numpy 与 list 之间的转换

- sklearn 中的 cosine_similarity（机器学习模组中计算两个句子的相似度的函数)

- jieba 将一个句子分割成一个个分散的单词

- list 的操作 (listname，listname.index(max(score_list)))

3）具体代码实现

a. 定义一个字典

这个字典非常重要：以这个字典为依据，将用户输入的句子与已知句子进行相似度匹配完成在一个字典的基础上，完成句子转成数组的工作

b. 将输入的句子转成数组：def get_vector(data)

其中将句子分成分散的单词，并将句子与字典一一比对，如句子中的单词在字典中则为 1，否则为 0

最后将一个人类看得懂的句子，转成由 0 和 1 组成的 array

c. 主程序

将用户通过 flask 网页输入的内容，与每一个常见问题循环进行相似度匹配：得到一个分数的 list

通过找出 list 中最大的匹配 score，来获得用户的问题与哪个常见问题匹配度最高。

d.完成的代码

```
from flask import Flask
import requests
import numpy as np
import jieba
from sklearn.metrics.pairwise import cosine_similarity
app = Flask(__name__)
word_vector_list = ["python","课程","助教","基础","方式","重点","有效期","周期"]

def get_vector(data):
    vector_list = []
    for i in word_vector_list:
        if i in list(jieba.cut(data)):
            vector_list.append(1)
        else:
            vector_list.append(0)
    print('vector_list',vector_list)
    return vector_list

@app.route("/greedyai/<data>")
def hello_world(data):
    FAQ = ["Python课程是线上课程还是线下课程","Python课程有助教吗","我没有基础应该从哪个课开始学","Python的学习周期是多久",
"Python课程的学习方式是什么呢","python学习重点是什么","Python课程的有效期是多久呢"]
    FQA_array=[]
    FAQ_array = list(map(get_vector,FAQ))
    score = []
    score_list = []
    s1_vector_list = get_vector(data)
    question_list_array = np.array(s1_vector_list).reshape(1,-1)
    for i in range(len(FAQ_array)):
        FAQ_test = np.array(FAQ_array[i]).reshape(1,-1)
        score = cosine_similarity(FAQ_test,question_list_array)
        scoretolist = score.tolist()
        score_list.append(scoretolist[0])
    maxvalue = max(score_list)
    index = score_list.index(max(score_list))
    if maxvalue ==[0.0]:
        index = 7
    if index == 0:
        return "线上课程为主"
    elif index == 1:
```

```
        return "为提高服务效率和质量，课程都配备专业的全职助教"
    elif index == 2:
        return "大周老师的Python基础集训营非常适合你哦，可以在这里学习：http://aijiaoai.greedyai.com/info/5"
    elif index == 3:
        return "如果你没有基础的话两个月可以搞定"
    elif index == 4:
        return "无需安装环境，在线直接写代码，看视频，看漫画，趣味性强"
    elif index == 5:
        return "练，只有反复地练习才能真的掌握"
    elif index == 6:
        return "有效期是1年，1年内可以无限次学习"
    else:
        return "哎呦，小主我不懂你的意思哎~"
if __name__ =='__main__':
    app.run()
```

五、 如何使用外部 API 实现聊天机器人

1. 需求

现在外面世界有很多 AI 的 Open API 供我们使用，我们要学会了解外部有什么 open API 供我们随意调用可以快速为我们的系统增加有用的功能。

2. API 介绍

1) 来源

图灵机器人：http://operation.tuling123.com/

2) API 接入说明

API V2.0 是基于图灵机器人平台语义理解、深度学习等核心技术，为广大开发者和企业提供的在线服务和开发接口。

目前 API 接口可调用聊天对话、语料库、技能三大模块的语料:

聊天对话是指平台免费提供的近 10 亿条公有对话语料，满足用户对话娱乐需求;

语料库是指用户在平台上传的私有语料，仅供个人查看使用，帮助用户最便捷的搭建专业领域次的语料。

技能服务是指平台打包的 26 种实用服务技能。涵盖生活、出行、购物等多个领域，一站式满足用户需求。

3）使用说明

a.编码方式:

UTF-8（调用图灵 API 的各个环节的编码方式均为 UTF-8）

b.接口地址： http://openapi.tuling123.com/openapi/api/v2

c.请求方式： HTTP、POST

d.请求参数： 请求参数格式为 json

e.请求示例

```
{
    "reqType":0,
    "perception": {
        "inputText": {
            "text": "附近的酒店"
        },
        "inputImage": {
            "url": "imageUrl"
        },
        "selfInfo": {
            "location": {
                "city": "北京",
                "province": "北京",
                "street": "信息路"
            }
        }
    },
    "userInfo": {
        "apiKey": "",
        "userId": ""
    }
}
```

3. 代码

```python
from flask import Flask
import requests
import json

app = Flask(__name__)

# 装饰器中为访问的路径
@app.route("/greedyai/<data>")
#@app.route("/")
def hello_world(data):
    url = "http://openapi.tuling123.com/openapi/api/v2"
    dara_param= {
        "reqType":0,
        "perception": {
            "inputText": {
                "text": data
            },
        },
        "userInfo": {
            "apiKey": "b802b298ef864f22aa394fd91ea9c95a",
            "userId": "13814042224"
        }
    }
    response = requests.post(url =url,json= dara_param)
    print("response>>>>",response.text)
    print(type(response.text))
    total_result = json.loads(response.text)
    print('total_result>>>>',total_result)
    result = total_result["results"]
    print("result>>",result)
    text = result[0]
    print("text>>>",text)
    text_result = text['values']['text']
    print('text_result>>',text_result)
    return text_result
if __name__ =='__main__':
    app.run()
```

运行结果:

结尾

其实在累积了 5 年多的 python 经验，进入 AI 的世界并不是特别难，因为 AI 也只是在 Python coding 的基础上学习新的模块，只不过这些模块与数学有很大的关系，所以需要学习统计学的一些知识，在具备统计学的知识的基础上，就可以做一些非常简单的 AI 的相关的 coding，所以我们永远不会停止学习的脚步，永远不会被时代抛弃!

基于三层结构的自动化测试数据准备技术研究

◆ 作者：杨志刚 谢彬 孙辉

摘要：测试数据准备是业务测试中的重要一环，测试数据准备的好坏直接影响测试的质量和效率。自动化测试对测试数据准备有着更高的要求，它要求测试数据一次性准备完成，而不允许分批准备。本文提出的基于三层结构的自动化测试数据准备方案，构建了测试数据资源池，将数据模型和数据实例进行了分离，测试数据可以自动抽取和校验，解决了环境变更，测试数据失效的问题。

一、测试数据准备意义和目标

1.1、测试数据准备意义

在项目进入正式测试之前，有一个重要的环节，就是业务的测试数据准备，测试数据准备在测试中起着决定性作用。测试数据准备是指对原始测试数据不断进行分析、验证和检查形成结构化数据，能够高效为测试项目提供合理、有效测试数据的过程，从而覆盖测试业务和测试边界，满足测试的完整性，一致性等要求。

1.2、测试数据准备目标

测试数据准备的目标，是要提供完全满足测试需求的测试数据。其中对测试数据准备需要达到数据质量好，覆盖范围广和准备效率高三个目标。数据质量好是指准备的数据要有较好的可用性，对数据的类别要细化到可供筛选的粒度；覆盖范围广是指准备的数据不仅要覆盖要测试的所有系统，理论上测试数据的准备需要达到测试流程的全覆盖，但覆盖范围广不一定要求测试数据的数量多，要根据等价类和边界值等模型筛选有效的数据类别；准备效率高是指积累的测试数据，要有较高的检索条件和标签，在使用时能够较快搜索到。

1.3、自动化测试数据准备

在自动化测试中，由于测试方式与手工测试不同，测试数据的准备方法也与手工测试中测试数据准备的方法稍有不同，自动化测试不允许测试数据分批准备，所有的数据准备必须一次性完成，从而保证自动化测试案例可以全量不间断执行，因此自动化测试对测试数据准备有着更高的要求。

二、三层结构的数据建模技术

为了更好的管理特定业务数据，提高测试数据使用效率，通过建立数据资源池，加强数据统一管理。我们采用一种数据管理的三层结构模型，该模型将测试数据的规则和实例数据进行了分离，从而提高测试数据的复用度。同时该模型能以独立的方式向测试提供数据实例。

模型自底向上分别为数据结构层、数据类别层和数据实例层。数据结构层及数据类别层存储特定业务数据的模型信息，数据实例层是根据模型信息从目标测试环境中获取实例数据。三层结构的数据资源池的结构示意图如下:

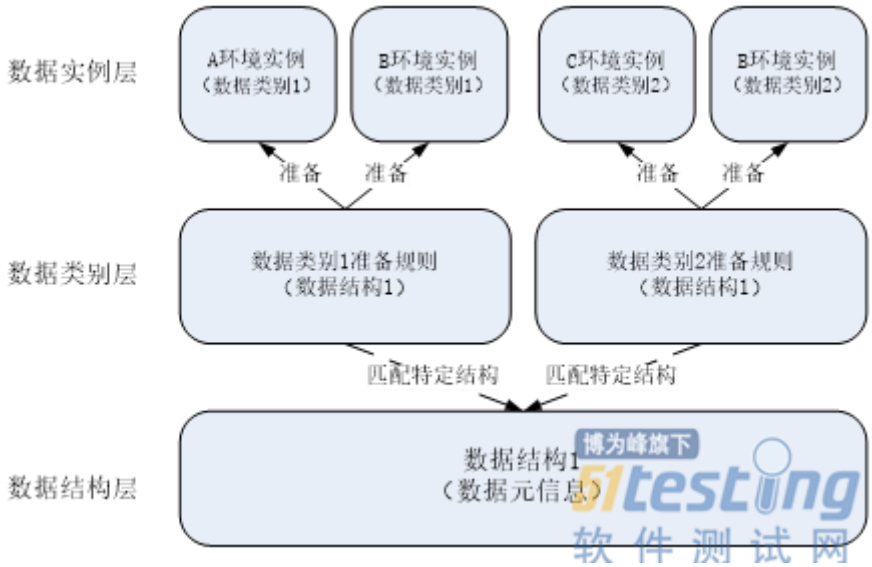


图 1 三层结构的数据资源池

模型的三个层次之间既相互独立又相互依存，最底层的数据结构层是对测试介质的抽象，保存着数据元信息，同一数据结构可以衍生多种数据类别；数据类别层，保存着测试数据的准备规则，并将不同的数据准备方式统一起来，按同样的标准管理，作为承上启下的中间层，是数据模型最重要的部分，也是数据模型中唯一与自动化测试案例交互的一层；最高层的数据实例层，是测试数据的展现层，同一数据类别可以在不同的测试环境生成不同的数据实例，保证数据类别的高可复用性，为功能和性能测试提供真实

可见的测试数据。三个层次共同构成了测试数据模型的整体体系。

2.1、数据结构层

数据结构层是三层结构数据资源池模型的底层，它是将基础测试数据资源进行抽象聚合，建立其相应的数据元信息，由此构成基础测试数据资源的数据结构模型。数据结构层主要是对测试介质的一种模拟，数据结构需要包含测试介质所需要的数据列，为了方便提供一致性的测试数据，还可以包含若干关联域，关联域不宜过多，必须和测试介质密切相关。

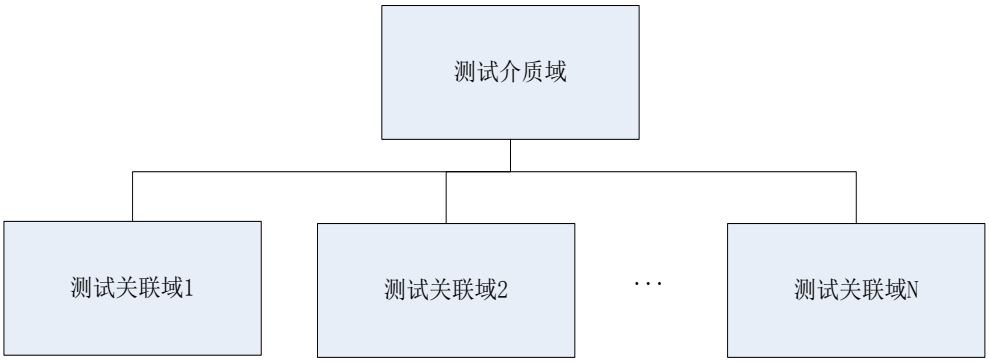


图 2 数据结构层域间关系

2.2、数据类别层

数据类别层是三层结构数据资源池模型的中间层，它是对测试数据的分类，将相同数据结构从业务角度细化成多个数据类别，每个数据类别所获得的数据具有相同的列，相当于对同一个测试介质进行不同类别的细化，并将其对应的实例数据准备规则模型化。数据类别是测试中积累的最重要数据资产，无论是针对手工测试还是自动化测试都有非常重要的意义。生成不同数据类别的依据可以是根据物理数据表中列的不同取值来进行划分，不同数据类别对应不同业务逻辑的测试介质，同时也可以根据自定义规则对数据类别进行生成，无论是根据数据表中的列，还是根据自定义规则，数据类别都可以将二者进行统一管理，从而使数据类别的生成方式对用户而言是透明的，降低数据类别的使用门槛。下图展示了测试中真实测试介质的数据类别。

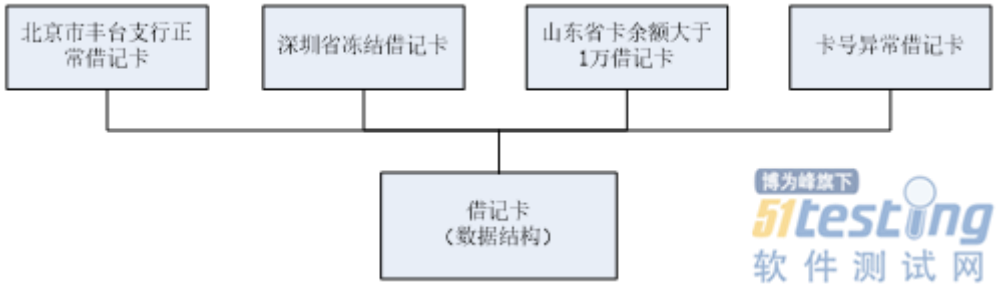


图 3 借记卡测试类别实例

最下面一层是借记卡的数据结构，它包含“卡号”，“省号”，“行部号”，“状态”，“余额”等若干列，所有数据类别的建立都基于这个数据结构。在此基础上建立了四个和借记卡相关的数据类别，代表测试中所需要的不同种类的借记卡，每个数据类别都保存着数据获取的规则，类别名称中由若干个描述标签组成，比如类别“北京市丰台支行正常借记卡”，“深圳市冻结借记卡”都是由“省市代码”，“卡状态”的标签组成，“山东省卡余额大于 1 万借记卡”是由“省市代码”，“余额”的标签组成，以上三种数据类别的抽取规则均可通过从后台铺底数据中抽取的方式实现。图中最后一个数据类别“卡号异常借记卡”，本身虽然不属于真实的借记卡，但在实际测试中也会应用到借记卡的相关测试场景，所以这种数据类别也应基于借记卡的数据结构建立，但它的数据获取方式有所不同，在被测系统的铺底数据里，几乎没有卡号异常的借记卡存在，因此这种数据类别的数据获取，采取自定义接口程序接入的方式，用户可以从接口程序库中选择，或者自行编写数据类别的生成算法。

描述标签，是数据类别最为显著的特征，它的全集是在数据结构中定义，标签值也需要在数据结构中预先定义。数据类别的本质就是对描述标签的不同子集进行组合，组合的方法要依据测试等价类和边界值理论，从而对同一数据结构下的测试数据进行分类，分类后的数据符合测试需求。描述标签的主要功能有两个方面，一是能够对数据类别进行限制，是生成数据类别的必要条件。本质上描述标签就是数据结构中的域，主域和附属域都可以作为描述标签，但并不是所有的域都要定义为描述标签，有些域只作为数据列显示即可，只有那些可以区别数据分类的域适合做成描述标签。二是使用描述标签可以快速对数据类别进行检索，提高测试人员查找测试数据的速度。

数据的准备规则存储在数据类别层，一般情况下，数据准备规则不会超过下表的四种类型。

表 1 数据准备规则类型表

序号	类型描述
1	从被测系统数据库中抽取，规则记录相应 SQL 语句（仅 select）
2	通过执行案例完成数据准备，规则记录案例号及提取数据信息
3	通过更新被测系统数据库或调用其关联系统，规则记录相应 SQL 语句（可为非 select）或外部关联系统调用信息
4	通过独立于被测系统的接口程序生成（准备规则与被测系统完全不相关），规则记录相应接口程序调用信息（逐步完善接口程序库）

数据结构层和数据类别层统称数据体系中的模型层，它是测试数据特征的抽象，用于描述数据的概念，它定义了数据间的完整性约束，并将约束关系通过建模的方式体现，实例数据是在模型层的基础上进行展现，并提供给测试人员使用。

2.3、数据实例层

数据实例层是三层结构数据资源池模型的顶层，它的主要功能是展现实例测试数据。与数据结构层及数据类别层不同，数据实例层是与被测系统后台环境相关的，其将数据类别层中特定数据类别在特定环境中完成实例数据准备、提取及存储。虽然测试案例关联的是数据类别，但在执行时，实际使用的是特定环境下的实例数据。图示是不同数据类别对应实际数据的举例。

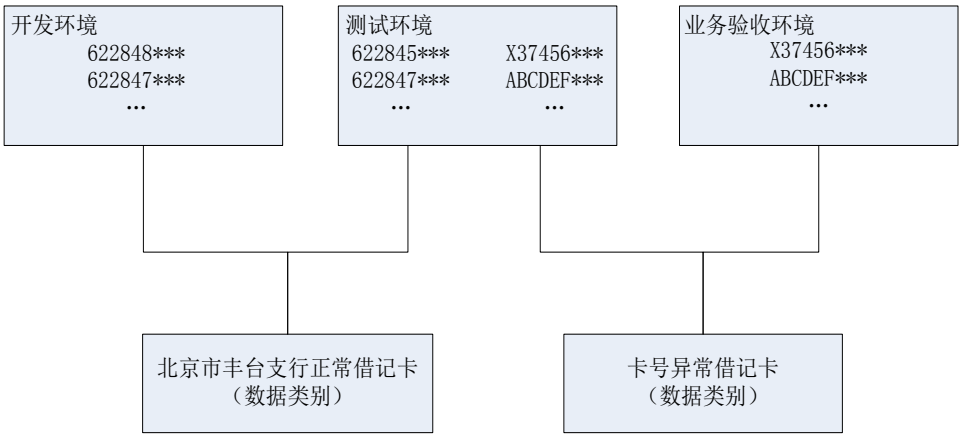


图 4 数据实例

三、基于模型的自动化测试数据准备技术

在实践中，上述三层结构的数据资源池模型与 ATP 自动化测试系统相结合，大大提高了数据资源池建设的自动化程度，提升了数据准备效率及质量，主要体现以下几方

面。一是数据池中数据自动化抽取，二是数据池中数据自动化校验，三是数据类别根据标签自动化生成。

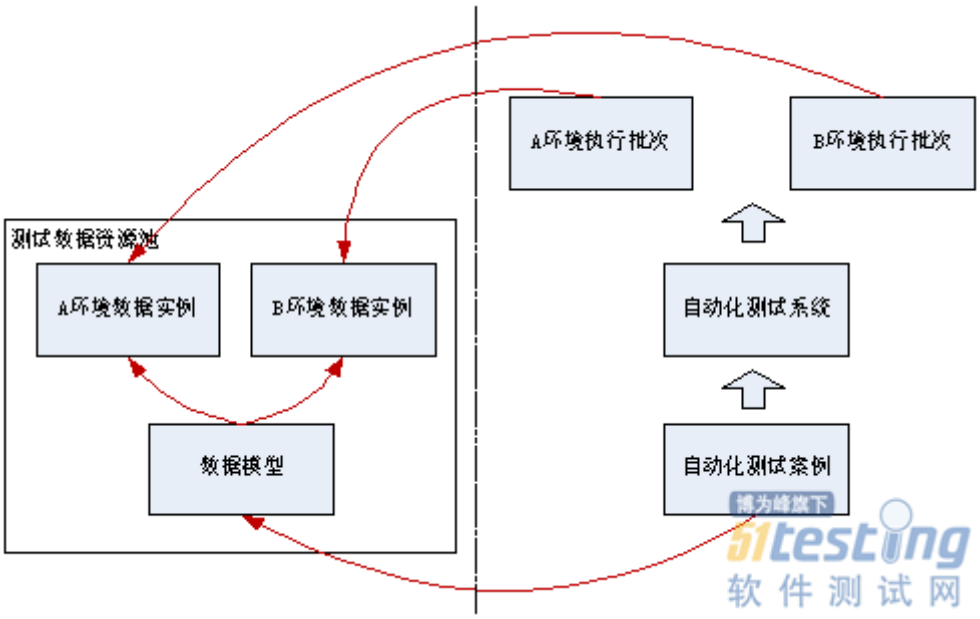


图 5 数据模型和自动化测试关系图

图中右侧是自动化测试执行的基本流程，自动化测试案例是自动化测试执行的最基本单元，一个案例下可以包含一个或若干个相关交易，交易间存在域值关联关系，即前置交易的输出结果可以作为后续交易的输入数据。批次是测试执行的容器，所有的案例都需要放入批次内执行，一个批下可以包含一个或多个案例，批次的结果可以作为测试资产永远保存，并在需要时进行回归测试。批次在执行时，可以动态选择执行的测试环境，同一个案例在不同测试环境下执行的结果一般是不相同的。

图中左侧是上文介绍的三层数据结构模型，其中图中的数据模型是由数据结构层和数据类别层共同组成，并且通过数据模型中保存的数据获取规则，在不同的测试环境中自动获取实例数据。

图中左右两侧的交互，也就是自动化测试系统和测试数据资源池的交互，主要体现在两方面。一方面是自动化测试案例设计时输入数据的静态配置，测试案例中的数据取值方式无非有三种。一是对于不变数据，填写成固定值，二是从前置交易的域值输出关联，三是由测试数据资源池中获取。通过测试数据资源池获取的方式，即是建立测试案例与数据类别的关联，使得被测试环境改变时，测试案例不失效。另一方面是自动化测试案例执行时输入数据的动态获取，批次中测试案例在执行时，通过动态查找关联数据类别中的实例数据，并对测试案例的输入域进行赋值，赋值的方式可以顺序进行也可以

随机选取。

四、总结

本文提出了一种基于三层结构的自动化测试数据准备方案，构建了测试数据资源池，将数据模型和数据实例进行了分离，数据准备规则保存在数据模型中。通过自动化技术，测试数据可以自动抽取和校验，提高了自动化测试数据的准备效率和复用程度。同时也为手工测试和性能测试的数据准备提供参考，具有一定的借鉴意义。

安卓 APP 兼容性测试 targetSdkVersion

◆ 作者：枫 叶

摘要： 随着移动网络的发展，手机已经成为老少皆用、必不可少的设备，手机 app 也被众人熟知。手机 app 的兼容测试显得有必要。这阵子并行进行着 3 项测试任务和自动化测试脚本编写，本想放弃本期的文章投稿，但在看到小编同学的友善提醒，还是争取写写近期关于安卓 app 兼容测试工作的实战总结。由于安卓手机的推陈出新，市场上安卓手机型号、安卓系统的丰富多样，在测试安卓 app 时候，除了功能性，同时最好用占据主流市场的各种真机进行兼容测试。

一、前言：

目前安卓市场上的主流机型主要有华为、小米、vivo、oppo、魅族、三星。安卓系统的名称颇有意思，似乎都跟甜食有关，让我们一起看看安卓的主要系统吧，从 Android 4.4KitKat（奇巧巧克力）、Android 5.0Lollipop（棒棒糖）、Android5.1 Lollipop（棒棒糖）、Android 6.0Marshmallow（棉花糖）、Android 7.0Nougat（牛轧糖）、Android 8.0Oreo（奥利奥），直到目前市场上手机已经有 Android 9.0Pie （派）的系统，目前 Android Q（正式名称是 Android 10）。安卓适配测试需要测试向上向下的兼容性。

二、了解一下向前兼容

当一个应用程序发布之后，可能没过几个月 Android 系统就发布了一个新版本，当然取决于具体的发布时间。这对 app 意味着什么，所有东西都不能用了吗？不会的，这里谈及向前兼容。它是 Android 非常关注的事情，即用户在升级到新版 Android 的时候，用以前版本的 SDK 构建的新应用应该不会出问题。由于 compileSdkVersion， minSdkVersion 和 targetSdkVersion 的作用，他们可以分别控制使

用哪些 API ，要求的 API 级别是什么，以及应用的兼容模式。

下图是 Android 各版本对应的 SDK 版本。笔者所在公司的安卓 app 支持 android4.4 及其以上的手机，个人觉得以目前的手机更新换代速度，老手机几乎没有用户在用，向下兼容到 4.4 足矣。

平台版本	SDK 版本	版本名称
Android 4.4 – 4.4.4	19 – 20	KitKat
Android 5.0 – 5.1.1	21 – 22	Lollipop
Android 6.0 – 6.0.1	23	Marshmallow
Android 7.0 – 7.1.2	24 – 25	Nougat
Android 8.0 – 8.1	26 – 27	Oreo
Android 9.0	28	Pie
Android 10.0	29	Android Q

三、Targetsdkversion

（An integer designating the API Level that the application targets. If not set, the default value equals that given to minSdkVersion.

This attribute informs the system that you have tested against the target version and the system should not enable any compatibility behaviors to maintain your app's forward-compatibility with the target version. The application is still able to run on older versions (down to minSdkVersion).

As Android evolves with each new version, some behaviors and even appearances might change. However, if the API level of the platform is higher than the version declared by your app's targetSdkVersion, the system may enable compatibility behaviors to ensure that your app continues to work the way you expect. You can disable such compatibility behaviors by specifying targetSdkVersion to match the API level of the platform on which it's running. For example, setting this value to "11" or higher allows the system to apply a new default theme (Holo) to your app when running on Android 3.0 or higher and also disables screen compatibility mode when running on larger screens (because support for API level 11 implicitly supports larger screens).

There are many compatibility behaviors that the system may enable based on the value you set for this attribute. Several of these behaviors are described by the corresponding platform versions in the Build.VERSION_CODES reference.

To maintain your application along with each Android release, you should increase the value of this attribute to match the latest API level, then thoroughly test your application on the corresponding platform version.

这是官方文档提到的定义。即 Targetsdkversion 指定应用目标的 API 级别的一个整数。如果没有设置，默认的值等于给予的最小 minSdkVersion。

这个属性通知系统已经针对目标版本进行了测试，系统不应该启用任何兼容性行为来维护应用程序与目标版本的正向兼容性。该应用程序仍然能够在旧版本上运行(下到 minSdkVersion)。

随着 Android 每个新版本的发展，一些行为甚至外观可能会发生变化。但是，如果平台的 API 级别高于应用程序的 targetSdkVersion 声明的版本，则系统可能启用兼容性行为，以确保应用程序继续以您期望的方式工作。可以通过指定 targetSdkVersion 来匹配它所运行的平台的 API 级别来禁用这种兼容性行为。例如，将这个值设置为“11”或更高，允许系统在运行 Android 3.0 或更高版本时向应用程序应用一个新的默认主题(Holo)，在运行较大屏幕时禁用屏幕兼容模式(因为对 API level 11 的支持隐含地支持较大屏幕)。

根据为该属性设置的值，系统可以启用许多兼容性行为。构建中相应的平台版本描述了其中的一些行为。以 VERSION_CODES 为参考。

要在每个 Android 版本中维护应用程序，应该增加这个属性的值以匹配最新的 API 级别，然后在相应的平台版本上彻底测试应用程序。

四、targetSdkVersion 是 android 向前兼容的主要方式

除非更新 targetSdkVersion，否则不会改变应用的行为。 这允许人们在处理行为更改之前使用新的 API。

Android 系统自 6.0 开始，提供动态权限机制，对于敏感权限（存储，定位，录音，拍照，录像等），需要在 APP 运行过程中动态向用户申请。

在 Android 上使用动态权限，要求 APP 编译的目标 sdk（即 targetSdkVersion）为 23 及以上，22 及以下系统会执行缺省处理（手机厂商也可能定制处理），APICloud 为简便开发，默认配置 targetSdkVersion 为 20，即走系统缺省处理，不允许更改。

有许多原生 APP 转到 APICloud 开发后，因 targetSdkVersion 降级而导致无法覆盖安装。从 2018 年 11 月开始，GooglePlay 要求 APP 编译目标 sdk 必须为 26 及以上，否则不予提交审核。

当设置的 targetSdkVersion 大于等于 23 时，即开启了动态权限，如果你的 APP 带有定位，录音，拍照，录像等敏感功能时（所有权限见文档），必须使用动态权限机制，先判断是否具有该功能操作权限，再进行操作，如果不具备相应的权限，对应的功能是失效的（也可能导致崩溃）。

当你设置的 targetSdkVersion 大于等于 23 时，如果是编译自定义 loader，安装到手机后，需要先在设置中给应用打开存储空间权限。否则，WiFi 同步后，loader 无法正常加载代码。

为保证动态权限尽可能适配更多厂商的手机以及顺利上线 Google Play，targetSdkVersion 目前推荐设置为 26，对应的系统为 Android 8.0。

五、 compileSdkVersion

compileSdkVersion specifies the Android API level Gradle should use to compile your app. This means your app can use the API features included in this API level and lower.

即 compileSdkVersion 指定了 Gradle 编译你的 App 时使用的 Android API 版本，你的 App 可以使用该版本或者更低版本的 API 特性。简单来说，如果你用了一些比较新的 API 特性，那么你的 compileSdkVersion 就不能低了。通常它应该是 Android 最新的稳定版本。比如 Android Pie(9.0)已经正式发布，那么我们的 compileSdkVersion 最好直接提升到 28（即 9.0 对应的版本号）。

六、minSdkVersion

An integer designating the minimum API Level required for the application to run. The Android system will prevent the user from installing the application if the system's API Level is lower than the value specified in this attribute. You should always declare this attribute.

Caution: If you do not declare this attribute, the system assumes a default value of "1", which indicates that your application is compatible with all versions of Android. If your application is not compatible with all versions (for instance, it uses APIs introduced in API Level 3) and you have not declared the proper minSdkVersion, then when installed on a system with an API Level less than 3, the application will crash during runtime when attempting to access the unavailable APIs. For this reason, be certain to declare the appropriate API Level in the minSdkVersion attribute.

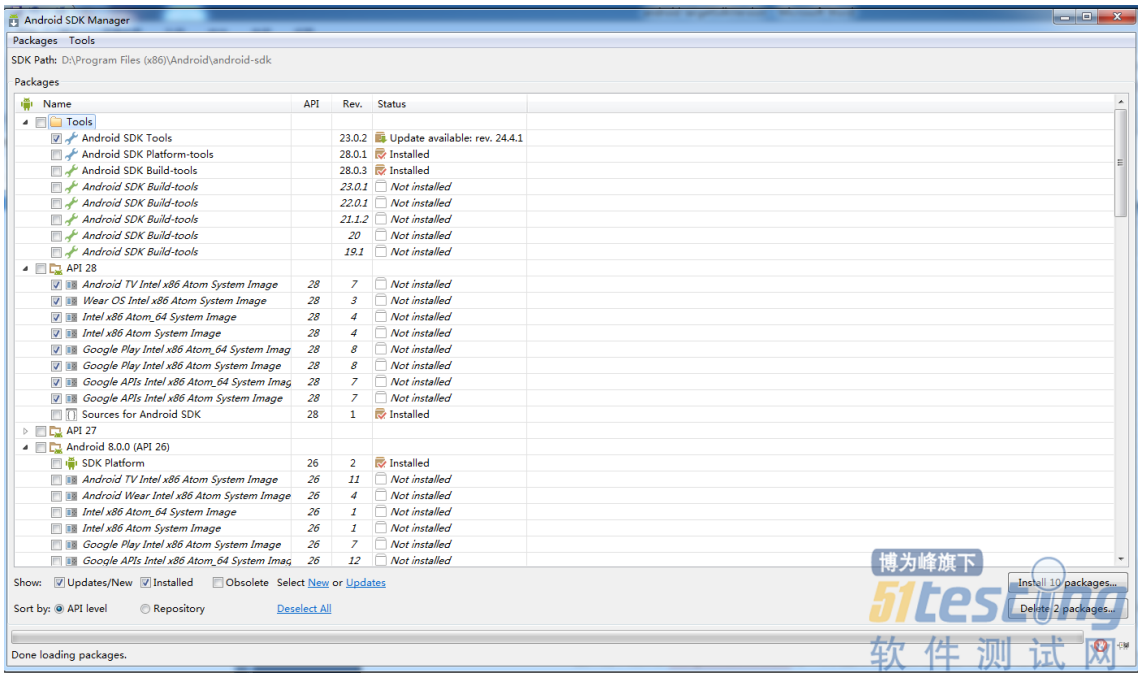
minSdkVersion 指定了 App 运行所需最低的 API 级别，比如你设置它为 23（Android 6.0），那么该 App 不能在低于 Android6.0 版本的设备上安装。从理论上来说，如果你设置 minSdkVersion 为 1，则可以最低兼容到最早的 Android 版本——然而，这就意味着你要为老设备的兼容做大量的工作，显然并不可取，毕竟从统计上来说已经没有人用那么老的设备了。

七、buildToolsVersion

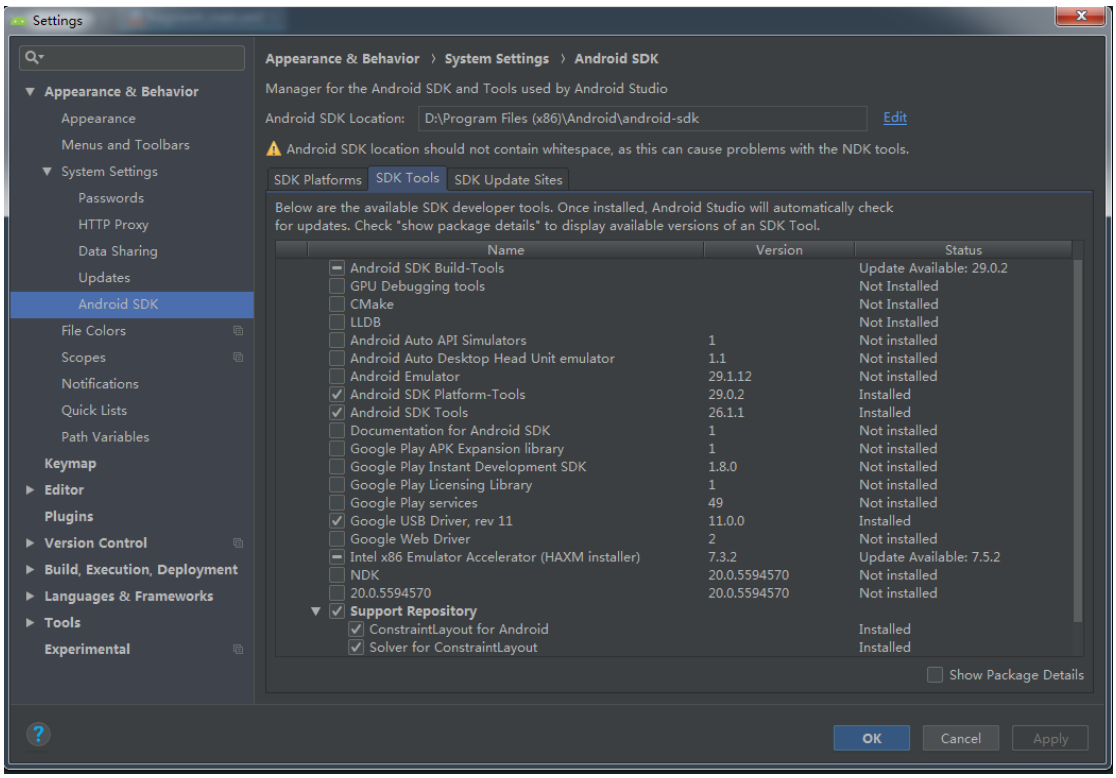
buildToolsVersion specifies the version of the SDK build tools, command-line utilities, and compiler that Gradle should use to build your app. You need to download the build tools using the SDK Manager.

即 buildToolsVersion 指定了 Gradle 在编译 App 时使用的 SDK build tools、命令行、程序、编译器等的版本，这些都是通过 SDK Manager 下载的。它的值实际上是一个字符串，在 Android Studio 里我们通过 Tools->SDK Manager，然后点选“SDK Tools”并勾选

“Show Package Details”就能看到本地电脑已经下载的 SDK Build-Tools 的版本。



一般来讲，选择跟 compileSdkVersion 同一个大版本下的最新稳定版本即可，如下图里这么多版本的 Build-Tools，如果在开发 App 时可以选择 29.0.2 版本。



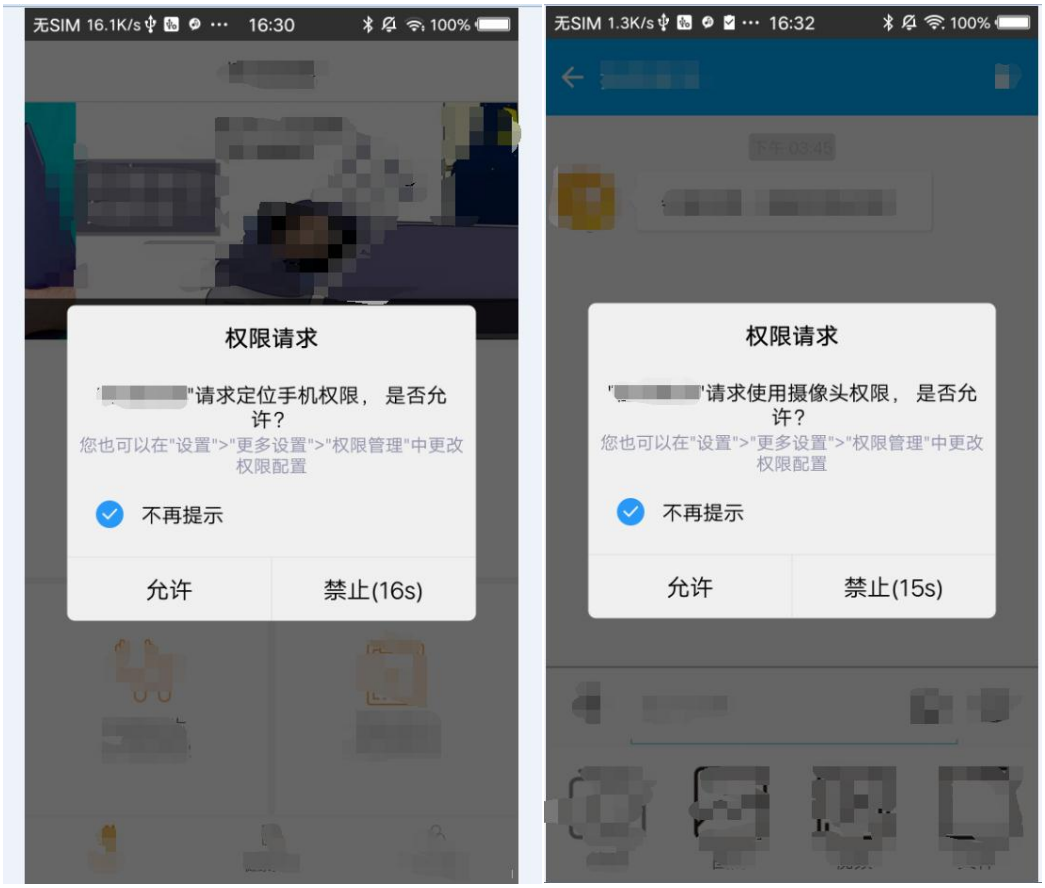
项目实况

项目的背景是 java 语言，开发环境是 android studio，跟兼容相关的有 2 大模块，一

个是个人信息中上传头像，另一个是环信聊天中发送图片、文件和视频。

开发人员送测时告知 targetsdkversion=26，因此测试人员需要考虑安卓 8 的手机的向上兼容，从目前市场手机的可行性出发，向上直到 android9。理论上向下到 Android 4.4.4，开发的应用程序支持 Android 4.4.4 及以上。但是对老设备的兼容做大量的工作，显然不可取，于是向下从 Android 7.0、7.1.2、Android 6.0、6.0.1 到 Android 5.0 – 、5.1.1。对这些系统对应的真机上进行 app 的测试，主要测试范围也是跟兼容相关的模块。

在手机的设置-权限管理-app 权限中，可以设置隐私安全（定位手机、访问手机识别码）和设备安全（使用摄像头、录音、打开/关闭 WLAN、打开/关闭蓝牙）。如果设为“总是询问”，那么在打开你们 app，会弹出窗口提示是否允许或禁止。



这是列举的其中两个页面，如果禁止，会影响到拍照功能的使用。允许之后，使用拍照功能，打开摄像头。结果出现了这样的报错信息:

Stack trace:

android.os.FileUriExposedException:

file:///storage/emulated/0/Android/data/com.chengyifamily.patient/chengyi%23oranger/oim0000008820/image/oim000000882015647093259 14.jpg exposed beyond app through ClipData.Item.getUri()

at android.os.StrictMode.onFileUriExposed(StrictMode.java:1799)

at android.net.Uri.checkFileUriExposed(Uri.java:2354)

at android.content.ClipData.prepareToLeaveProcess(ClipData.java:832)

at android.content.Intent.prepareToLeaveProcess(Intent.java:9499)

at android.content.Intent.prepareToLeaveProcess(Intent.java:9484)

at android.app.Instrumentation.execStartActivity(Instrumentation.java:1525)

at android.app.Activity.startActivityForResult(Activity.java:4399)

at android.support.v4.app.BaseFragmentActivityApi16.startActivityForResult(BaseFragmentActivityApi16.java:54)

at android.support.v4.app.FragmentActivity.startActivityForResult(FragmentActivity.java:67)

at android.support.v4.app.ActivityCompat.startActivityForResult(ActivityCompat.java:152)

at android.support.v4.app.FragmentActivity.startActivityFromFragment(FragmentActivity.java:798)

at android.support.v4.app.FragmentActivity$HostCallbacks.onStartActivityFromFragment(FragmentActivity.java:907)

at android.support.v4.app.Fragment.startActivityForResult(Fragment.java:1028)

at android.support.v4.app.Fragment.startActivityForResult(Fragment.java:1017)

at com.hyphenate.easeui.ui.EaseChatFragment.selectPicFromCamera(EaseChatFragment.java:992)

at com.hyphenate.easeui.ui.EaseChatFragment.requestPermission(EaseChatFragment.java:729)

at com.hyphenate.easeui.ui.EaseChatFragment.requestFilePermission(EaseChatFragment.java:740)

at com.hyphenate.easeui.ui.EaseChatFragment.access$000(EaseChatFragment.java:76)

at com.hyphenate.easeui.ui.EaseChatFragment$MyItemClickListener.onClick(EaseChatFragment.java:640)

at com.hyphenate.easeui.widget.EaseChatExten

……

原来从 Android 7.0 开始，谷歌收回了访问文件的权限，就是一个应用提供自身资源

文件给其它应用使用时，如果给出 file://xxx 这种格式的 URI 的话，谷歌会认为目标应用不具备访问此文件的权限，便会抛出 FileUriExposedException 的异常。

到此，可以把问题跟开发同学沟通，以便于他们进行修复工作。

你是如何汇报工作的？——软件测试的价值论述

◆作者：侯 峥

最近的心情有些浮躁，遇到的一些事情动摇了我的职业观念。测试的价值是什么？在我原本的观念中，测试的作用是保证项目质量，是非常重要的职位。我的老师也曾经说过，对于项目的各个阶段，最不能够压缩的就是质量保障环节的时间。但是，近段时间我遇到的问题却让我不得不对这句话的权重性有所怀疑。我从不否认软件测试环节在项目中的重要性，但我对软件测试环节在整个项目中的权重性持保留意见，这也体现出了很多人对于测试的偏见和误解。出现这种想法的起因有二，一是部门经理削减测试组补充人员名额；二是产品组的同事说“为什么要有测试环节，开发自测不就行了”。对于第一个原因，我暂时可以勉强理解为今年的大环境不是很乐观，领导需要控制预算或者说领导有其他的战略布局；对于第二个原因，我听到之后心里很是难受，觉得自己的努力被否定了，也许我不应该在意这样的话，做好自己的本质工作才最重要。非常欣慰的是，公司开发组的同事还是认可测试的存在意义，开发同事说“有测试这样的岗位存在就必定有其存在的意义，不能因为你个人的不理解就否定整个行业”。

“存在即有意义。”在面对别人的质疑的时候，最先爆发出来的情绪不应该是愤怒，或者是哀伤。而是应该理智的去面对，去据理力争。软件行业发展这么多年，软件测试岗位的出现和发展正是对软件测试岗位认可性最强有力的证据。面对外行的质疑，我们应该用测试人员的专业价值和业绩来反击。都 9102 年了，作为软件测试行业的一份子，我们需要更加努力的普及软件测试的重要性，展现软件测试在整个项目过程中的重要性。**那么软件测试的价值需要如何体现？或者说是如何高效的体现？**

首先，大家普遍认知的是软件测试是保证项目质量的一个重要环节。这个价值是普遍意义上的，也是其表象上的价值。但是，实质上软件测试的价值不仅仅于此，其价值在项目进度、项目成本，甚至企业口碑方面都有间接的体现。软件测试的普遍价值意义已经是老生常谈，这里不再过多的叙述，本文在软件测试的潜在价值方面讨论如何体现软件测试的价值，即软件测试存在更深层的意义。

要体现是价值光靠描述是不够的，要学会转化。把测试的价值转化成数字会更容易理解，在展现形式上图表化会使得表象上更具有冲击性。

一、软件测试在项目进度方面的价值

有很多时候会出现这样的误区，项目迟迟不能上线是因为还有 Bug 没有修改。这个时候就会有人跑过来问测试，“怎么测出这么多 Bug？还能不能上线了？”每次遇到这样的问题我都很困惑，所以 Bug 是我造成的咯？即使我不测试，那个 Bug 仍然存在，只不过没有被显现出来而已。心里虽然这样想，又不能表现出来，只能微笑着回答：“Bug 没有改完建议不要上线。”这是项目中很常见的现象，看似测试环节影响的项目上线，其实根本不是这么回事。

软件测试的一条真理是 Bug 发现的越早，修复的成本越低。测试的工作不仅仅是在项目开发完成才介入，而是在项目立项初期就开始介入。其间参与需求评审、设计评审、测试用例设计、测试用例评审、测试项目、项目上线。在各个阶段从测试的角度提出意见，将 Bug 扼杀在萌芽中，减少修复 Bug 花费的时间，避免由于设计不合理导致的返工。这样就要求测试人员具有较高的专业素质及经验，要求测试人员掌握架构、中间件特点、开发语言特点、交互设计、用户体验等方面的知识，有丰富的项目经验。我们在汇报工作的时候可以从这些角度来体现软件测试对于项目进度的影响。

二、软件测试在项目成本方面的价值

最初软件测试设置的意义就是为了减少项目成本，软件测试的一条真理是 Bug 发现的越早，修复的成本越低。但是从公司运营的角度上来看，技术人员本就是成本，而测试人员更是成本中的成本，从管理人员的角度考虑，如果想要削减人员成本必定会从测试人员下手。与此同时，随着软件行业的成熟以及各种开发工具、框架和插件的普及，项目开发产生的 Bug 似乎不再那么的多。而由于开发语言的特点或者开发人员的经验导致的 Bug 也在一定程度上由于框架和插件的存在得以规避。那么软件测试人员存在而减少的 Bug 修复成本似乎就不是那么的明显。但是，实际上项目在上线之前存在的 Bug 数量并没有由于技术的成熟而减少太多，甚至由于技术的问题 Bug 数量呈现上升趋势。所谓“成也萧何，败也萧何”。

从管理人员的视角入手，即测试人员最擅长的用户体验和换位思考。相较于项目的质量，管理人员更加关注的是“钱”的数量。在汇报工作或者是项目总结的时候，不要

再单纯的统计项目存在的 Bug 数量及对应级别，要学会依据 Bug 级别和对应级别的数量将数字转换成金额（参考样式见表 1），让管理人员直观的看到测试人员替公司省了多少钱，与测试人员的成本比较，他的花费是值得的。当然，金钱不是衡量价值的唯一要素。

表 1 Bug 数量统计表

Bug 优先级	Bug 数量	影响权重	金额转换（￥）
P1			
P2			
P3			
P4			
P5			
总计（￥）			

三、软件测试在企业口碑方面的价值

口碑是衡量价值的另一个要素。软件测试的本质是保障软件的质量，产品质量是企业的根本，直接影响企业的口碑。一个企业只有拥有好的口碑才能良好的运作下去。软件测试保障项目质量，进而可以帮助企业提升品牌价值。就像很多人喜欢买名牌，除了名牌是财力的体现之外，名牌产品的质量也的确有保证。当然，并不排除一些小众品牌也有好的质量，但是那只是极个别的。用户通常会通过口碑来快速的筛选自己想要的东西，企业之间的合作亦是如此，拥有好口碑的企业更加容易接触到优秀的资源，吸引优秀的人才。我们在汇报工作的时候，把软件测试在这方面的影响量化直观的体现出来，能够使得管理人员更加意识到软件测试的重要性，软件测试不只是保证项目质量，更是企业口碑的保证。

另外，Bug 是不能够被完全测试出来的，软件测试只能尽可能多的发现 Bug 这是公认的事实。而且由于市场的原因，有些 Bug 即使没有被修复完成，项目仍然需要上线。在这种情况下，我们需要对未修复的 Bug 进行评估，对于必须解决的 Bug 坚持修复完成后上线，做好最后一道防守；对于可以延后修复的 Bug 制定解释口径及容错方案，将 Bug 的影响转化成为营销战略。比如：ofo 小黄车的“Bug 营销”、微博的“宕机营销”。这需要软件测试人员具备出技术之外的能力，比如：行业预测、市场营销、公关处理等。

表 2 已解决 Bug 统计表

Bug 类别	Bug 数量	影响范围
UI—页面显示		
UE—用户体验、易用性		
FC—功能问题		
PF—性能问题		
IF—接口问题		
CK—用户操作提示信息问题		
BF—程序打包问题		
AL—程序算法错误问题		
RC—与需求不符		
RRC—需求变更		

表 3 未解决 Bug 统计表

Bug 类别	Bug 数量	影响范围	后续方案
UI—页面显示			
UE—用户体验、易用性			
FC—功能问题			
PF—性能问题			
IF—接口问题			
CK—用户操作提示信息问题			
BF—程序打包问题			
AL—程序算法错误问题			
RC—与需求不符			
RRC—需求变更			

其实体现测试价值的最核心要素就是实力，也就是找出最多的 Bug，引导开发最快的解决；其次才是一些体现价值的套路和方法。几年前由于测试受到偏见，工作中很难推动测试工作，我写下了《测试中的“曲线救国”》这篇文章，里面主要的思想还是要

加强自身实力，同时争取同事的认可。几年后的今天在经历了许多事情之后，遇到同样的问题我解决问题的侧重点已经有所转变，今天我解决问题的方式大多采取一些场面上的手段，不知这样的方式对还是不对。也许是在挣扎了这么多年之后的经验，就好比我解决前几年的困难时，最有效的方式是换了家公司，非常感谢我现在的公司让我知道了我还很重要。尽管如此，当资源紧缺的时候，最先被削减的还是测试的资源。

测试的定位决定了我们无法扭转别人的思想，也没有能力控制别人的言论。也许某一天等我做到决策层位置的时候，我才能真正的为测试争取一些权益。现在我能做的就尽量做好，剩下的交给老天爷吧。最近也在考虑转行的事情，但是还是希望离测试这个岗位不要太远，希望有一天我有能力把资源向测试倾斜，而不是需要费力的去争取。也许这才是真正的“曲线救国”吧。最喜欢那句话：“愿你出走半生，归来仍是少年”。

高效的移动应用程序测试项目的三大策略

◆译者：枫 叶

当手机应用程序市场在扩大，开发人员努力地尽快将他们的产品带上市场。结果，QA 团队比以往在更大的压力下节约利用他们的资源，而且更快地、更准确地交付他们的应用程序的测试。

团队可能要处理成千上万的测试人员，他们需要在测试过程中得到指导。在手机应用程序发布之前，获得关于其功能的反馈显得尤为重要。但是同样重要的是，测试团队要学会如何去降低应用程序测试的成本和代价。

考虑到资源效率，这里有三种方法来管理移动测试项目。

1、考虑主要的测试挑战

我和我的团队已经对移动应用程序进行了一段时间的测试，我们已经意识到移动环境特有的测试挑战。为了确保最终产品的高质量和性能，测试团队需要考虑这两个主要方面。

首先是严重的设备碎片。与桌面或 web 应用程序不同，移动应用程序将被用于各种设备和平台。不要忘记操作系统也有许多不同的版本。移动设备上的碎片对开发者来说是一个很大的挑战。开发人员想要创造同一应用程序的不同版本，而且要保证它在给定的不同的操作系统版本上运行正确。QA 部门需要记住，这些操作系统有不同的兼容性，兼容性使获得应用程序的安全和管理变得更加地困难。

移动测试工具和资源也需要考虑到。由于工具的可用性，我们经常决定外包选定的测试区域。考虑到多设备遵从性，先进的测试工具和有效的方法对于内部测试团队来说是不可用的。这就是为什么如此多的组织选择通过与测试伙伴的合作来引入人才和工具。

由于测试预算没有增加，并且最后期限比以往任何时候都要紧迫，那些决定测试过

程将会是什么样子的人需要认真考虑外包测试活动。外包可能是一个不错的选择，因为它为开发人员提供了获得合格的专业人员和测试工具的途径，但是需要一段时间才能找到一个可以信任的合作伙伴。

根据我们的经验，外包某些测试活动可以极大地帮助降低测试成本。大多数情况下，我们选择外包不重要的应用元素，并适当地专注于内部应用程序的核心活动。

2、决定模拟器还是真机

在测试策略中另一个重要的决定是关于选择模拟器还是真机。

在开发的早期，我们经常使用设备模拟器，因为它们允许我们快速地测试应用程序的不同元素，并且在敏捷开发环境中良好地工作。假如你考虑测试基本的应用程序功能性，模拟器无疑是成本效益好的，而且很完美。我们在开发应用程序特性时也经常使用设备模拟器。

但是这并不意味着我们在完全避免使用真机。经验告诉我们，在一台模拟器上进行测试就可以发布一个成功的应用程序是不可能的。如果你不能在实际设备上测试你的应用程序，你将永远无法真正了解它的活动在现实生活场景中是如何发挥作用的。只有物理设备允许你测试各种因素，比如不同的电池状态，从 Wi-Fi 到 4G 的多种网络，或者网络密度。物理设备也给开发对用户如何与应用程序交互以及应用程序如何在不同设备上工作一览无余。显然这种类型的测试环境不能在模拟器上重新创建。

因此，以高效和低成本的方式测试应用程序的最佳策略是将设备模拟器和真机混合使用。恰当的平衡取决于您的 QA 团队需要什么。

3、安全的 beta 测试者

这就引出了 beta 测试和 beta 测试人员的话题。明智的做法是在已经忠实于您的产品的用户中寻找 beta 测试人员，例如在线表单中的人员或在社交媒体上与你的公司进行交流的人员。根据我的经验，识别经常与应用程序交互的用户的最佳方法之一是通过帮助台。这些人通常是积极参与你的应用程序的人，他们中的许多人会非常乐意成为你下一个版本的 beta 测试人员。

一些公司利用众筹平台为他们的应用寻找 beta 测试者。他们在 alpha 阶段准备应用程序，并将其发布在 Kickstarter 等平台上，以提高知名度，并可能获得更多资金。

另一种获得一批客观的 beta 测试人员的方法是有限的发布。将你的应用程序发布到一个特定的国家或地区，可以让你在广泛发布前测试应用程序的新功能。

然而，如果你的应用程序不包括监控或分析工具，那么选择最好的 beta 测试人员并不能保证得到好的结果。通过这些工具，你就可以跟踪用户与应用程序的交互方式，并检查哪些功能运行不正常。

聪明的做法是在你的应用程序中加入这个功能，让你能够开启或关闭不同的应用程序功能。通过这种方式，你可以让一组 beta 测试人员测试一个特性，而另一组测试另一个特性。这是获得应用程序在野外测试时执行情况的真实视图的最佳方法，而且这比要求用户报告他们的活动或填写调查更有效。

开发移动应用程序的测试策略意味着要面对该领域的多个挑战。但是通过使用上面这三种策略，你可以在你的移动应用程序发布之前得到关于其功能的有价值的反馈，同时还能记住时间和预算。

API 测试该了解的一些技术细节

◆ 作者：Erin

首先，我们想，什么是 API?

简单来说，API，是应用程序接口（Application Programming Interface，又称为应用程序编程接口），是软件系统不同组成部分衔接的约定。一个软件系统越庞大，需要用到的接口相对越多，同时接口的复杂度和接口的设计都需更好的设计和提升。

那么，什么是 API 测试？API 测试其实是一种用程序或工具来发送数据，同时验收系统的返回值的方法。这种测试更偏向于业务实现逻辑。常见的网络协议有 TCP、Http、webservice、socket 等，http?和 webservice 都是基于 TCP/IP 协议的应用层协议，webservice 是基于 http 的 soap 协议传输数据。我们今天主要说最常见的基于 http 协议的 API 的测试。

一般来说，API 测试是除去单元测试和白盒测试之外最能够从底层发现问题的测试方法。那么，API 测试需要注意哪些技术细节呢？换句话说，怎么做一个好的 API 测试呢？

我们从以下四点说起:

1、用例设计

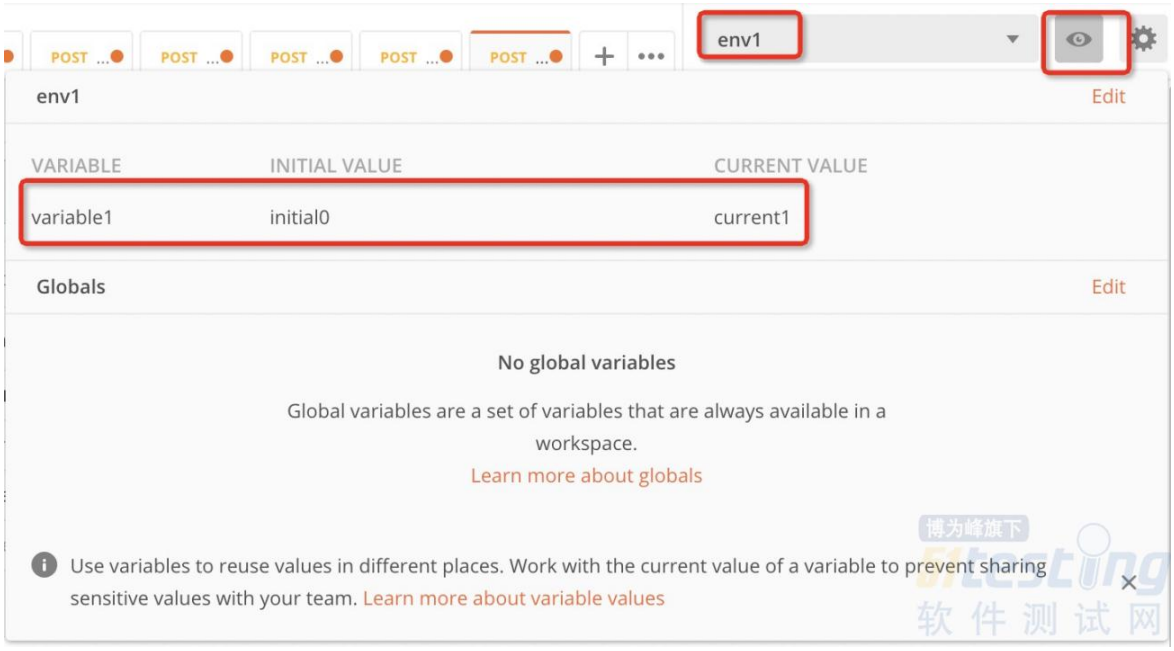
如果把 API 看做一个黑盒的话，那么我们首先可以设计基于边界值法、等价类划分法等的黑盒用例，这些设计思想其实占据很大成分。常见的比如参数值的边界，参数缺失/多余，参数空/非空，特殊字符等；对于复杂的参数，比如结构体/数组链表等，可以考虑其最大长度限制/内置特殊字符等。其次，请求方式/不合法的数据格式/不合法的 cookie 也会影响到一个接口的返回值。还有，有些接口涉及到加密解密，需要传一些密钥值，一些非合法密钥的检验，来观察 API 的响应情况。最后，如果手里有很详细的接

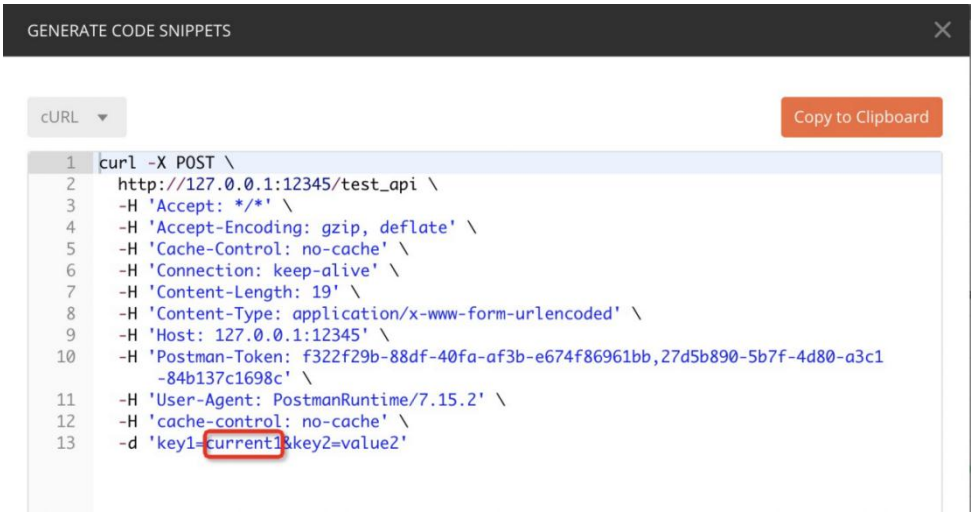
口文档，把每个 return code 都覆盖到，很有必要。比如正常是 200 OK，此外还有 400(不合法请求)，401(未授权)，429(太多请求)等，或其它一些自定义的 error code，覆盖的过程，也是把工程代码分支覆盖的过程。

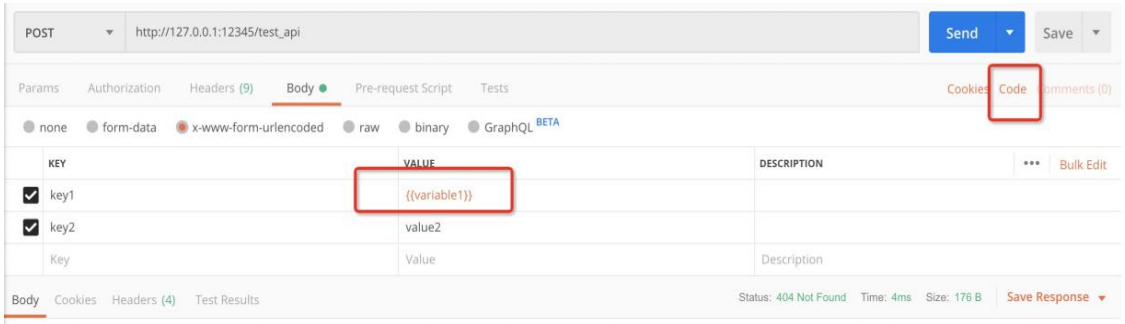
2、请求工具

一般用 Chrome 浏览器的话，postman 的使用频次应该是最多的了。也可以下载 postman 等位客户端。之前用 Firefox 浏览器的时候，还用过 HttpRequester。不管用哪种，方法一样。首先，填写好测试 URL，选择测试方法(GET/POST/PUT/DELETE)，设置 Header(常用的有 content-type/Cookie 等)，设置 authorization type(Basic Auth 等)，最后在 body 中填充测试数据。接下来，点击 send 就好啦，这样就发送了一个请求到你的目标 API 了。

这里面比较复杂的可能就是 body 了，常用的格式有 form-data, x-www-form-urlencoded, applictaion/json 等。用哪种格式，正规的接口文档里开发同学就会注明。此外，postman 有一个功能很 nice，就是它可以配置环境变量。把配置信息抽象成类，不同环境对应不同的实例，初始化设定后，在 request 请求中通过实例成员变量来引用不同的值，从而在需要的时候通过切换环境来选择不同的配置信息。比如：我配置了一个 env1 环境，并添加了一组 key 和 value，那么当我引用{{}}这个变量时，就会替换成你所配置的。







3、程序设计

首先是选择用哪种语言和相应的请求包，比如 python 的话常用的就是 requests 库，而 Golang 的话你可以使用 net/http 包或是 gorequest 包。拿 python 来说，短短十来行代码就可以模拟发送一个请求。

```
import requests

url = "http://127.0.0.1:12345/test_api"

payload = "key1=key1&key2=value2"
headers = {
    'Content-Type': "application/x-www-form-urlencoded",
    'Host': "127.0.0.1:12345",
}

response = requests.request("POST", url, data = payload, headers = headers)

print(response.text)
```

但是难点并不于此。

如何组织用例？

一般来说，用例可以分为单个接口用例和场景用例，单个接口很简单，针对一个接

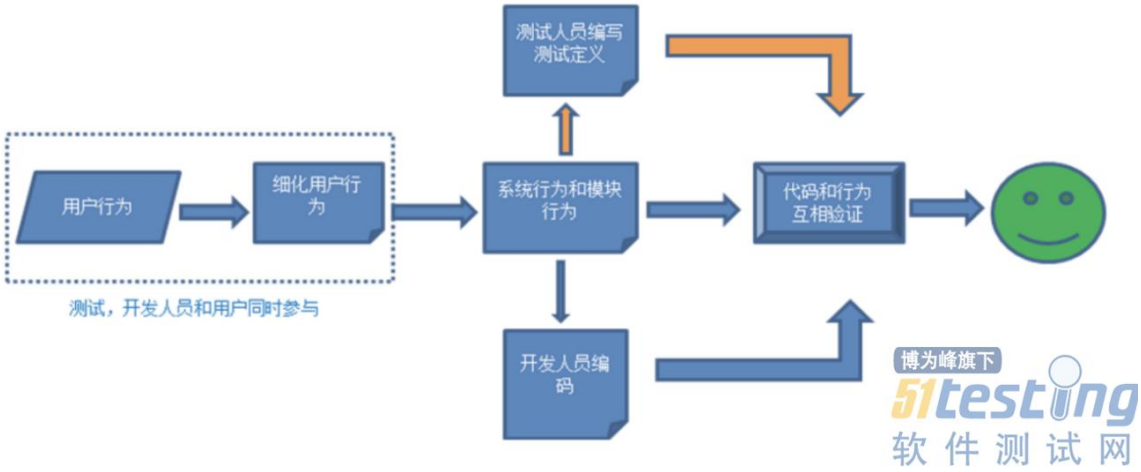
口的参数设置边界值，对应校验不同的返回；场景用例涉及到多个接口的调用，用以简单模拟用户行为。

测试数据应该放在哪里？

测试数据可以放在用例里，也可以放在 json/yaml 等配置文件中。如果涉及到图片/视频等比较大的测试文件，最好不要存在本地，可以在测试服务器搭建一个简单 server，比如使用 ngnix，接下来只需要访问这些测试文件的链接就好了。

使用哪种风格的测试框架？

现在基本有 BDD 和 TDD 两种框架之分，我更倾向于使用 BDD，也就是“行为驱动测试”。这个风格有两个优点：1.场景分级，易读清晰，方便定位失败的用例 2.新手好上手，BDD 的过程就是写 case 的过程。下面是它的一个流程图。



4、接口分析

如果你的团队里面，能够维护一份完整细致的接口文档，当然是最好的。如果没有的话，那么，优先去推动开发生成一份合适的 API 说明文档吧。如果达不到这个阶段，但是也要做自动化，那么你就要掌握基本的抓包分析工具，能够自己去抓包分析形成 API 文档。

所以接口分析是很必要的，也属于接口测试的高阶提升。比如接口定义是否冗余，接口的请求字段是否冗余，接口调用是否得到了所有期望的信息，接口调用是否合理方便，接口是否做到最数据库进行了正确的变更，接口的平均响应速度是否在可接受范围等。这些指标的分析，有时候可以反馈给开发同学，对优化整体性能也是有益处的。同时，这些分析可以帮助你更好理解这个过程的来龙去脉，理解这些 API 做了什么，又产

生了什么影响。

总之，API 测试上手很简单，但做得好，做成工程化还真需要费一点力气，一些技术细节的把控和提升，会无形中提升整体的测试水准；而如何让 API 测试真正在我们的日常工作中发挥出最大作用，也需慢慢研究和调整的。

Python 多线程在爬虫中的应用

◆ 作者：咖啡猫

题记： 作为测试工程师经常需要解决测试数据来源的问题，解决思路无非是三种：1、直接从生产环境拷贝真实数据 2、从互联网上爬取数据 3、自己用脚本或者工具造数据。前段时间，为了获取更多的测试数据，笔者就做了一个从互联网上爬取数据的爬虫程序，虽然功能上基本满足项目的需求，但是爬取的效率还是不太高。作为一个精益求精的测试工程师，决定研究一下多线程在爬虫领域的应用，以提高爬虫的效率。

一、为什么需要多线程

凡事知其然也要知其所以然。在了解多线程的相关知识之前，我们先来看看为什么需要多线程。打个比方吧，你要搬家了，单线程就类似于请了一个搬家工人，他一个人负责打包、搬运、开车、卸货等一系列操作流程，这个工作效率可想而知是很慢的；而多线程就相当于请了四个搬家工人，甲打包完交给已搬运到车上，然后丙开车送往目的地，最后由丁来卸货。

由此可见多线程的好处就是高效、可以充分利用资源，坏处就是各个线程之间要相互协调，否则容易乱套（类似于一个和尚挑水喝、两个和尚抬水喝、三个和尚没水喝的窘境）。所以为了提高爬虫效率，我们在使用多线程时要格外注意多线程的管理问题。

二、多线程的基本知识

进程： 由程序，数据集，进程控制块三部分组成，它是程序在数据集上的一次运行过程。如果同一段程序在某个数据集上运行了两次，那就是开启了两个进程。进程是资源管理的基本单位。在操作系统中，每个进程有一个地址空间，而且默认就有一个控制进程。

线程： 是进程的一个实体，是 CPU 调度和分派的基本单位，也是最小的执行单位。它的出现降低了上下文切换的消耗，提高了系统的并发性，并克服了一个进程只能

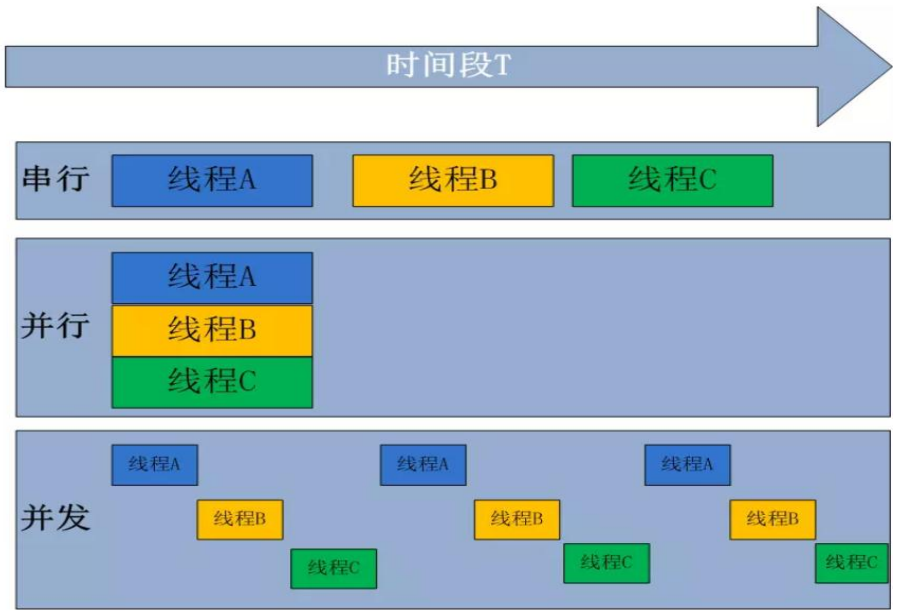
干一件事的缺陷。线程由进程来管理，多个线程共享父进程的资源空间。

进程和线程的关系：

- 一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程。
- 资源分配给进程，同一进程的所有线程共享该进程的所有资源。
- CPU 分给线程，即真正在 CPU 上运行的是线程。

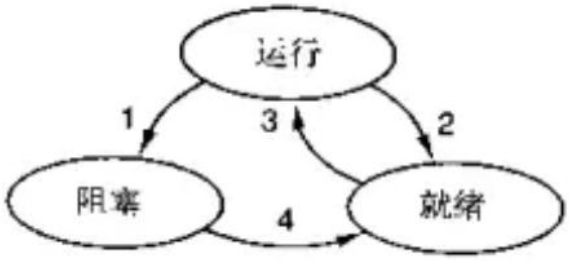
线程的工作方式：

如下图所示，串行指线程一个个地在 CPU 上执行；并行是在多个 CPU 上运行多个线程；而并发是一种“伪并行”，一个 CPU 同一时刻只能执行一个任务，把 CPU 的时间分片，一个线程只占用一个很短的时间片，然后各个线程轮流，由于时间片很短所以在用户看来所有线程都是“同时”的。并发也是大多数单 CPU 多线程的实际运行方式。



进程的工作状态：

一个进程有三种状态：运行、阻塞、就绪。三种状态之间的转换关系如下图所示：运行态的进程可能由于等待输入而主动进入阻塞状态，也可能由于调度程序选择其他进程而被动进入就绪状态（一般是分给它的 CPU 时间到了）；阻塞状态的进程由于等到了有效的输入而进入就绪状态；就绪状态的进程因为调度程序再次选择了它而再次进入运行状态。



1. 进程为等待输入而阻塞
2. 调度程序选择另一个进程
3. 调度程序选择这个进程
4. 出现有效输入

三、多线程通信实例

还是回到爬虫的问题上来，我们知道爬取博客文章的时候都是先爬取列表页，然后根据列表页的爬取结果再来爬取文章详情内容。而且列表页的爬取速度肯定要比详情页的爬取速度快。

这样的话，我们可以设计线程 A 负责爬取文章列表页，线程 B、线程 C、线程 D 负责爬取文章详情。A 将列表 URL 结果放到一个类似全局变量的结构里，线程 B、C、D 从这个结构里取结果。

在 PYTHON 中，有两个支持多线程的模块：threading 模块--负责线程的创建、开启等操作；queque 模块--负责维护那个类似于全局变量的结构。这里还要补充一点：也许有同学会问直接用一个全局变量不就可以了么？干嘛非要用 queue？因为全局变量并不是线程安全的，比如说全局变量里（列表类型）只有一个 url 了，线程 B 判断了一下全局变量非空，在还没有取出该 url 之前，cpu 把时间片给了线程 C，线程 C 将最后一个 url 取走了，这时 cpu 时间片又轮到了 B，B 就会因为在一个空的列表里取数据而报错。而 queue 模块实现了多生产者、多消费者队列，在放值取值时是线程安全的。

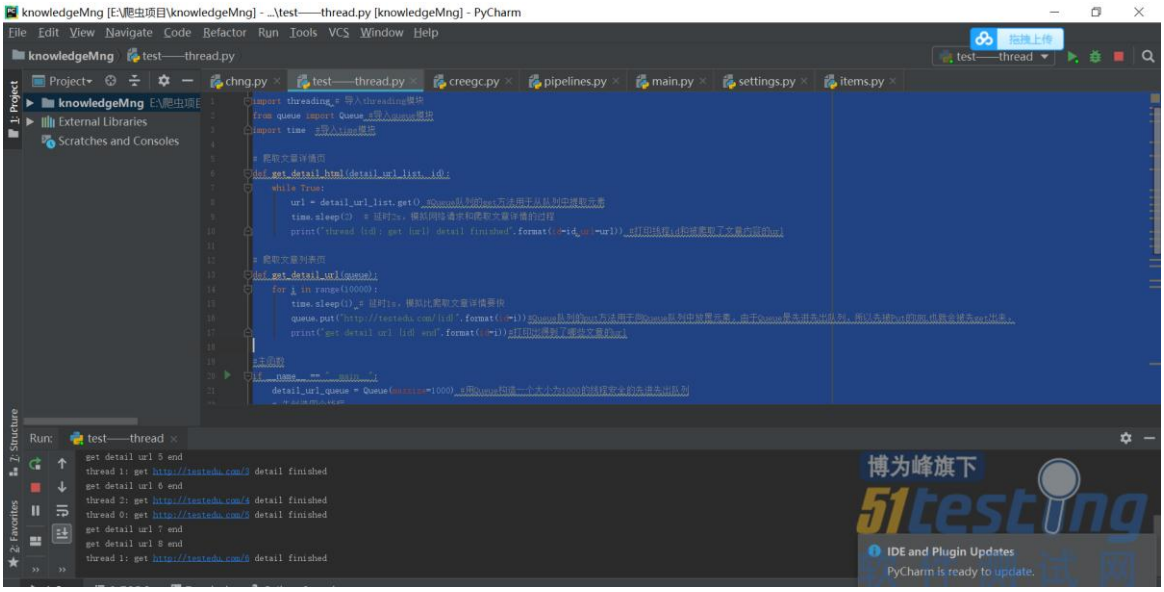
废话不多说了，直接上代码给大伙看看：

```
import threading #  导入 threading 模块
from queue import Queue #导入 queue 模块
import time   #导入 time 模块


# 爬取文章详情页
def get_detail_html(detail_url_list, id):
```

```
        while True:
            url = detail_url_list.get() #Queue 队列的 get 方法用于从队列中提取元素
            time.sleep(2)  # 延时 2s，模拟网络请求和爬取文章详情的过程
            print("thread {id}: get {url} detail finished".format(id=id,url=url)) #打印线程 id 和被爬取了文章内容的 url


# 爬取文章列表页
def get_detail_url(queue):
    for i in range(10000):
        time.sleep(1) # 延时 1s，模拟比爬取文章详情要快
        queue.put("http://testedu.com/{id}".format(id=i))#Queue 队列的 put 方法用于向 Queue 队列中放置元素，由于 Queue 是先进先出队列，所以先被 Put 的 URL 也就会被先 get 出来。
        print("get detail url {id} end".format(id=i))#打印出得到了哪些文章的 url


#主函数
if __name__ == "__main__":
    detail_url_queue = Queue(maxsize=1000) #用 Queue 构造一个大小为 1000 的线程安全的先进先出队列
    # 先创造四个线程
    thread = threading.Thread(target=get_detail_url, args=(detail_url_queue,)) #A 线程负责抓取列表 url
    html_thread= []
    for i in range(3):
        thread2 = threading.Thread(target=get_detail_html, args=(detail_url_queue,i))
        html_thread.append(thread2)#B C D  线程抓取文章详情
    start_time = time.time()
    # 启动四个线程
    thread.start()
    for i in range(3):
        html_thread[i].start()
    # 等待所有线程结束，thread.join()函数代表子线程完成之前，其父进程一直处于阻塞状态。
    thread.join()
    for i in range(3):
```

```
        html_thread[i].join()

    print("last time: {} s".format(time.time()-start_time))#等 ABCD 四个线程都结束后，在主进程中计算总爬取时间。
```

运行结果:



后记：从运行结果可以看出各个线程之间井然有序地工作着，没有出现任何报错和告警的情况。可见使用 Queue 队列实现多线程间的通信比直接使用全局变量要安全很多。而且使用多线程比不使用多线程的话，爬取时间上也要少很多，在提高了爬虫效率的同时也兼顾了线程的安全，可以说在爬取测试数据的过程中是非常实用的一种方式。希望小伙伴们能够 GET 到哦！

一个好测试，首先得是一个好管理

◆ 作者：黎晓萌

十多年测试经验，多年的测试管理工作下来，一个很大的感悟是：想要做好测试，要想让自己负责的项目质量信得过，首先要做的就说要做一个好管理。有些公司也把测试叫质量管理，可见这个岗位对“管理”的要求。

那么，如何才能做一个好管理，去管理好自己负责的项目呢？我给大家分享几点自己的感悟。

自我管理

做一个好的管理者，首先得是一个好的自我管理者。自我管理者，意味着对自己的严格要求，严格要求自己的工作态度，严格要求自己的细心程度，严格要求自己的进取心。

工作态度，是你对待工作认真与否的体现。

有良好的工作态度，你就会去认真学习你所负责的业务，了解清楚业务背后的技术框架，和产品经理去请教业务特性的细节，和开发工程师了解技术实现的细节。

有良好的工作态度，你就会去思考你工作过程中有哪些地方可以改进，时常审视自己测试工作中有哪些测试工具还使用的不够熟练，学习测试工具的实现原理、应用场景和无法实现的地方。

有良好的工作态度，你就会去反思每一次版本迭代后，项目研发过程中有哪些不足可以去优化和改进，可以通过哪些手段和度量去改进研发过程。

细心程度，是你对被测产品细致程度的体现。

越是细心，你越能发现测试用例背后可能存在的遗漏与不足，察觉兼容性场景中缺少的某个容易忽视的场景。

越是细心，你越愿意举一反三的改进当前用例给你带来的新思考，愿意和开发一起走查一个缺陷背后可能还有的更多隐患。

越是细心，你越能从一个异常测试场景发现更多功能实现方式背后可能的风险，结合技术架构、实现方式去发现更多开发工程师和产品经理没有考虑到的细节。

进取心，是你对自己严格要求的体现。

充分的进取心，会让我们做测试中多去尝试新的工具和方法，去挖掘工具背后的东西，去总结新方法的优点与不足，形成自己的测试体系，沉淀出有自己特色的方法论。

充分的进取心，会让我们在测试过程中补齐自己能力的短板，无论是基础能力项，还是新的技能要求，我们都会有对自己更高的要求，去为了“更高、更快、更强”的目标优化自己的测试手段和测试工具。

充分的进取心，会让我们在遇到难题时有更坚定的意志去坚持、去深挖，在遇到技术瓶颈时去思考更好的解决方法，在遇到效率问题时尝试更多的思路和方法去优化自己的测试脚本。

总之，对自己要求得越严格，自己就会在自我成长上收获更多别人无法教给你的东西。

他人管理

对他人对管理，有对上管理，也有对合作伙伴的管理，有对开发工程师的管理，也有对产品研发过程中其他环节同学的管理。

对上管理，要求我们对上级交代给我们对工作有很好的掌控，安排下来的事情，第一时间作出响应，适时给出自己的解决方法和方式，定期汇报自己解决处理的进度和状态，对过程中的风险和困难及时知会。

对伙伴的管理，要求我们勇于承担边界工作，对伙伴工作中可能的不足和缺失进行补位，对伙伴做得比自己好的地方给予鼓励和肯定，对伙伴给自己的帮助真诚地表示感谢。

对开发工程师的管理，要求我们和开发工程师处理好关系，不要把关系变得对立，他们是对被测项目技术实现最了解的人，多从他们那了解被测项目的技术细节，多向他们请教测试方法与测试方案，多跟他们沟通缺陷产生的原因和修改方法，都会对我们测

试项目有积极的帮助。

对流程中其他同学的管理，要求我们和大家一起做产品质量的守护者，和大家一起发现项目流程过程中的问题，和大家一起商讨问题的解决方式，提出自己对产品特性的理解和认知，与大家一道让被测项目变得更易用、体验更佳、问题更少。

项目管理

能做到对项目进行管理，说明我们的能力已经有了一定的积累，做团队内也得到了认可，我们就需要进一步发挥我们“质量管理”的价值和作用。

研发过程中质量数据的度量。通过对千行代码缺陷率、缺陷存活时间、缺陷一次性修复成功率、高中低级缺陷比例等维度的考察和审计，去把握被测项目的研发质量情况，结合项目的迭代状态和团队的状态，有针对性的进行优化改进，是自己作为一名质量管理人员的基本工作。

研发过程中项目风险的把控。通过对研发过程中各个阶段时间的消耗、修复缺陷的效率、测试计划的完成准时度，测试周期内缺陷的走势、发布前一周缺陷的收敛情况，产品发布后质量情况等风险指标进行把控，提高自己对项目风险的控制力，让项目质量中自己的可控范围之内，做真正的项目质量管理者。

团队管理

当做到了这一步，相信我们的能力也被更广泛的人认可和赞赏。我们就需要帮助更多的伙伴成长，让大家一起享受作为项目测试人员应有的尊重和价值。

让每一个人都得到成长，不仅是被测业务的了解，还有更多测试方法、测试技能的掌握，让大家一起跟着项目的前进有所收获，得到更多的认可与尊重。

让每一个人都有目标感，知道自己所做的一切都是为了什么，为了实现这个目标该怎么样做，做的过程中需要了解什么、掌握什么，做完这一切我能收获什么。有了这样的指引，每一个人都会少一些迷茫，多一些努力，去和大家一起实现自己的价值。

让每一个人都能承担起自己该有的责任，这是我们成长最快的方式。一份责任背后，我们会有更强的自我驱动力，把这份责任做好，我们就会有更大的成长。

总之，想要做好测试，想要做好质量管理这份工作，我们就要多多学习管理知识，帮助自己更好的成长。

银行线上信贷系统接口自动化测试探索

◆ 作者：王海林 王师孟

近年来，伴随着金融科技、互联网金融业务的浪潮，依托大数据、人工智能、云计算等技术的进步，线上信贷产品在银行信贷业务中占据越来越多的比重，承载这些业务的银行系统通常具有关联系统众多、接口调用关系复杂等特点，测试难度较大。于是，我们以接口自动化测试为切入点，对线上信贷业务的测试工具及其应用开展探索和研究。

一、工具选择

工欲善其事，必先利其器。我们调研了 Postman、Poster、Jmeter 等几款开源的自动化测试工具：

JMeter: 一款开源的性能测试工具，操作简单、方便，既有 JDBC request 操作数据库数据，也有 Http request 和 Soap request 应对测试。它通常可以用于测试静态或者动态资源的性能（文件、Servlets、Perl 脚本、java 对象、数据库和查询、ftp 服务器或者其他的资源），使用 JMeter 提供的图形化界面分析性能指标或者在高负载情况下测试服务器/脚本/对象的行为。

Poster: 火狐浏览器自带的借口测试工具，插件中安装即可，界面简单明了，容易上手。

Postman: 原为谷歌浏览器的扩展工具，目前已成为独立软件，界面简洁，与 Poster 差别不大，功能较 Poster 更为强大，可保存执行场景并分类管理。支持通过 Newman 等插件进行命令行提示符调度运行。

Cucumber: 与以上几种特定工具不同，更像是一个以场景编辑器，需与其他测试

工具结合使用。功能齐全，需要一定编程技术基础，学习成本较高。

经过调研，我们发现，Postman 是一款功能较为强大的 Http 调试与模拟插件，可以发送几乎所有类型的 Http 请求。同时，它界面功能较为友好，使用方便，应用较为广泛。

二、线上信贷业务接口测试的实际应用

完成工具调研之后，我们结合线上场景贷款的典型业务流程，对 Postman 工具进行了应用。

线上贷款产品，通常由客户自助发起，从通过各种渠道入口（例如：掌上银行、网上银行、网点柜面）进行申请和办理。针对我们关注的线上贷款运算逻辑，我们使用 Postman 工具来模拟从各个渠道系统发送过来的请求报文，实现对信贷系统的测试，通

过调整不同的输入信息，验证系统处理逻辑。

在贷款申请、审批、放款三个环节，使用 Postman 编制三支接口功能测试脚本；在线上贷款的申请、审批和发放三个环节中，通过在 pre-request script 设置序列号进行参数配置，从返回值中抓取变量的返回结果进行变量存储关联，实现了不同环节的报文串联；通过在 Collections 集合中统一管理和调度脚本，使脚本集合能够支持业务从头到尾的全流程自动化测试和验证。

线上信贷业务系统除了关注业务处理逻辑之外，对于客户信息，如：姓名、身份证号、手机号等需要做重点的校验，包括唯一性校验、信息字段本身的合法性校验等。这无疑给我们的接口自动化测试带来了很大困难，对于这种多样性问题，Postman 有什么好的解决办法吗？**通过探索实践发现，我们可以编写 JS 脚本去解决这个问题。例如：**

1、客户姓名自动生成

```
// 自动化生成随机姓名
function getName() {
    var familyNames = new Array("赵", "钱", "孙", "李", "周", "吴", "郑", "王", "冯", "陈");
    var givenNames = new Array("子璇", "淼", "国栋", "夫子", "瑞堂", "甜", "敏", "尚", "国贤","胜利");

    var i = parseInt(10 * Math.random());
    var familyName = familyNames[i];

    var j = parseInt(10 * Math.random());
    var givenName = givenNames[i];

    var name = familyName + givenName;

    return name;
}
getName()
"李夫子"
getName()
"钱淼"
```

2、客户手机号码自动生成

```
> //自动化生成随机手机号
  function getMobleNumber(){

      var prefixArray = new Array("130","131","150","151","138","170","189");
      var i = parseInt(10 * Math.random());
      var prefix = prefixArray[i];

      for (var j = 0;j < 8;j++){
          prefix = prefix + Math.floor(Math.random()*10);
      }
      return prefix
  }
  getMobleNumber()
< "15000329625"
> getMobleNumber()
< "13168365451"
```


经过实际使用，Postman 工具可以较好的支持上述场景的接口测试，支持测试数据和测试参数的配置，统一管理和调度测试执行，通过脚本的导入导出实现脚本备份和共享。另外还可以通过使用环境变量配置实现同一测试集合脚本在不同测试环境中的运行，使用 Newman 插件实现脚本的命令行执行，大大提高了脚本实际应用的便利性和可扩展性。

《51 测试天地》（五十五）下篇 精彩预览

马上阅读