
目录

(四十八期 上)

接口测试填坑的那些事儿.....	01
Python 调用安卓 adb 命令 (下篇)	25
测试人员如何把控项目进度.....	33
如何做可达性测试.....	38
高级信息系统项目管理师考试经验总结.....	43
知道这些 , 轻松处理临时任务.....	46
愿有岁月可回首 , 且以勤奋共进步.....	49

接口测试填坑的那些事儿

◆ 作者：晴空

前言：谨以此文记录总结自己在实施接口自动化测试过程中遇到的坑和填坑过程，希望对做接口测试的同学有些许帮助~ 紧记一点：方法总比困难多，只为成功找方法，不为失败找借口!!!

一：实施背景

版本迭代快！QA 缺人手！UI 自动化 ROI 低！单元测试技术要求高，QA 做不动！有没有同感？亲们。

其实，分层自动化测试(UI+接口+单元测试)的理论大家都懂，只是真到实施推进的时才发现效果和预期差的不是一点儿半点，赤裸裸“理想很丰满，现实很骨感”的节奏。

先来看下 UI 自动化吧，尽管框架设计的好，使用 PageObject 模式或者 PageFactory 模式和各种封装，但是你怎么扛得住 UI 层的快速变动，有些团队对此的应对方案是跑 UI 自动化脚本之前先用爬虫爬去页面自动更新页面元素，嗯~ 这很 OK！然而 UI 自动化脚本的执行效率呢？又有同学说 我们用多线程(进程)来执行 UI 自动化测试脚本，这个点 get 的不错。

我想问的是：UI 自动化测试框架和测试用例以及测试数据的维护需要占用多少人力/工时资源？而 UI 自动化测试的产出又有多么可观？

当然，我们并不是不做 UI 自动化测试，我们这边 UI 自动化测试仅仅是覆盖主干业务的新建、查询、更改、删除，维护的总用例数不到 100 个。

接着，我们来看单元测试，单元自动化测试执行速度是快(秒杀 UI 自动化执行)，可以快速发现底层问题，问题是有多多少 QA 或者美其名曰的“测试开发”同学有实施单元



测试的能力？

这个是一个因素，另一方面是 你要做单元测试，前提是你得先明白开发同学写的方法是干什么用的，这无疑对 QA（或者测试开发）的时间资源占用极大。综合各种因素，个人对单元测试实施的建议是：QA 同学去设计用例，而单元测试脚本开发和执行由开发同学们去做。

最后，估计好多负责招聘的同学们都深有感触的一点是“好的 QA 同学太难招”，想干仗，首先你得有兵马有粮草。

那么妥妥的，把目标转移到接口测试来，接口自动化测试执行效率高(那也是轻松碾压 UI 自动化的执行)能很快的收到测试执行结果反馈，并且接口自动化测试维护比 UI 自动化简单，而技能要求又没有单元测试高，毫无疑问，接口自动化测试是最好突破口

PS: 这里小小地透露一点：听话，出活儿(产出高)，执行力强(即时反馈)的同学谁不喜欢？

二：技术选型

如果团队使用的开发语言是 Java，个人强烈建议接口自动化框架也使用 Java 语言来开发，常见套路是 Java+TestNG(Junit4)+HttpClient 这样，如此做法的好处是：

1: 活跃团队氛围，你用 Java 开发测试框架，Java 开发同学会更有话题和你沟通交流，彼此赶脚很亲切有木有~；

2: 后续拓展去实施单元测试，你也可以逐渐轻易地看懂被测试对象，设计出更优秀的测试用例。

有些测试团队里，大家普遍技术不强，我还是推荐 RobotFrameWork 这个框架，做接口自动化测试也是溜的飞起。

我们测试组的同学普遍使用 Python，那就没啥好说的了。

很多同学使用 Python 自带的测试框架 Pyunit，断言呢也是使用自带的，然后脚本执行使用 HTMLTestRunner，测试报告嘛，无疑是 html 格式。

我个人不建议使用 Pyunit 了，因为有更好用 nose 框架啊~

至于 html 格式的报告，更是不在话下 nose-html-reporting, nose_htmloutput 玩起来，如果你对 Python-Web 开发中的模板引擎 Jinja2 规则熟悉，那再好不过了，自己可以自定义测试报告格式，来个定制化报告是不是更炫^_^



还有断言，我不想执行过程中断言失败就中断后续的用例执行，所以选择是 `assery.py` 这个三方库来断言，`assert` 库里的软断言 `soft_assertions` 谁用谁知道 哈哈~别说我没告诉你哦

所以，我的选择是 Python+Nose(超越 `pyunit` 的测试框架)+`nose_htmloutput`(自定义测试报告格式)+`Json` 文件(管理接口用例数据)+`assert.py`(简单好用的断言库)

三：知识体系科普

接口自动化测试 ROI(投入产出比)虽好，首先我们需要掌握其关联的知识体系。

如果东一榔头西一棒子很容易形成知识点的碎片化，当真正去落地实施的时候反而受困。

我这里抛出问题，希望有机会看到此文的同学具体的答案在实践中充实

3.1: 接口的定义

先弄明白什么是接口，接口有哪些类型。

搞清楚被测对象才能更好的去做测试工作，不要害怕，恐惧是因为陌生!!!

3.2: http/https 协议

http 的报文格式，请求头，请求体。请求返回报文的格式，报文体。

接口设计的模式 `webservice/restful`。

PS: 重要声明----不要以为只有 http 协议接口，其他底层协议也可以多看看。

3.3: 测试用例设计

其实接口用例设计用到的方法比较常见。

对单个接口来说，就是用等价类，边界值方法，此外需要考虑的是参数缺失，重复操作等。

我负责任地告诉大家，接口测试的最大价值在于对业务场景的覆盖，所以~~~接口间的依赖是必不可少的，这时候设计用例就需要用到场景法了。

3.4: Python 基础知识

这个就无需费言啦，一种编程语言的基础语法(变量，条件判断，循环体等等)是首



先要掌握的。

此外，很多不错第三方库推荐给亲们~

- Excel 文件处理: openpyxl
- 日期处理: Arrow
- http 请求处理: requests
- 断言处理: assert.py
- 测试框架: nose
- Html 格式报告文件生成: nose_htmloutput

除此之外还有很多哈~ 后续会和同学们分享个人在学习大数据测试过程中的一些总结~

四：环境搭建

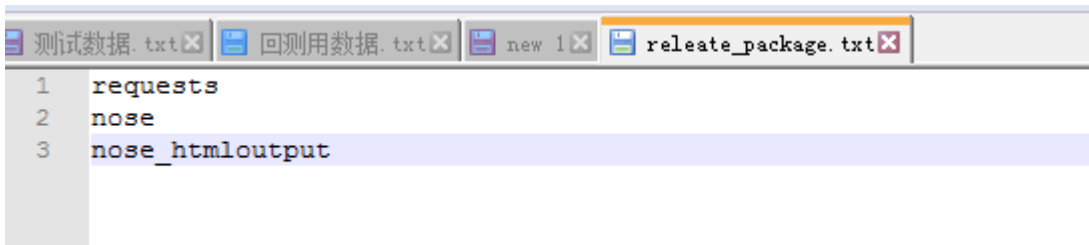
4.1: 包管理工具 pip

先说一点吧，很多同学还在犹豫使用 python2.x 版本还是 3.x 版本，我的个人建议是：不要犹豫毫无疑问地选择 3.x 版本。

配置 Python 环境的步骤我就不写了，如果这个都不会或者不愿意去 Google 下，那么请不要继续往下看这篇文章了，我也强烈建议这样的同学别做软件测试工作了。

编辑器的话，个人推荐 JetBrains 公司出品的 PyCharm，当然这个看个人喜好。

安装第三方库的话可以手动一个一个安装，还有一种方案是将所有的库文件名写在一个文件中，示例如下：



```

1 requests
2 nose
3 nose_htmloutput
  
```

```
pip install -r D:\releate_package.txt
```

执行结果如下图，因为我已经全部安装了

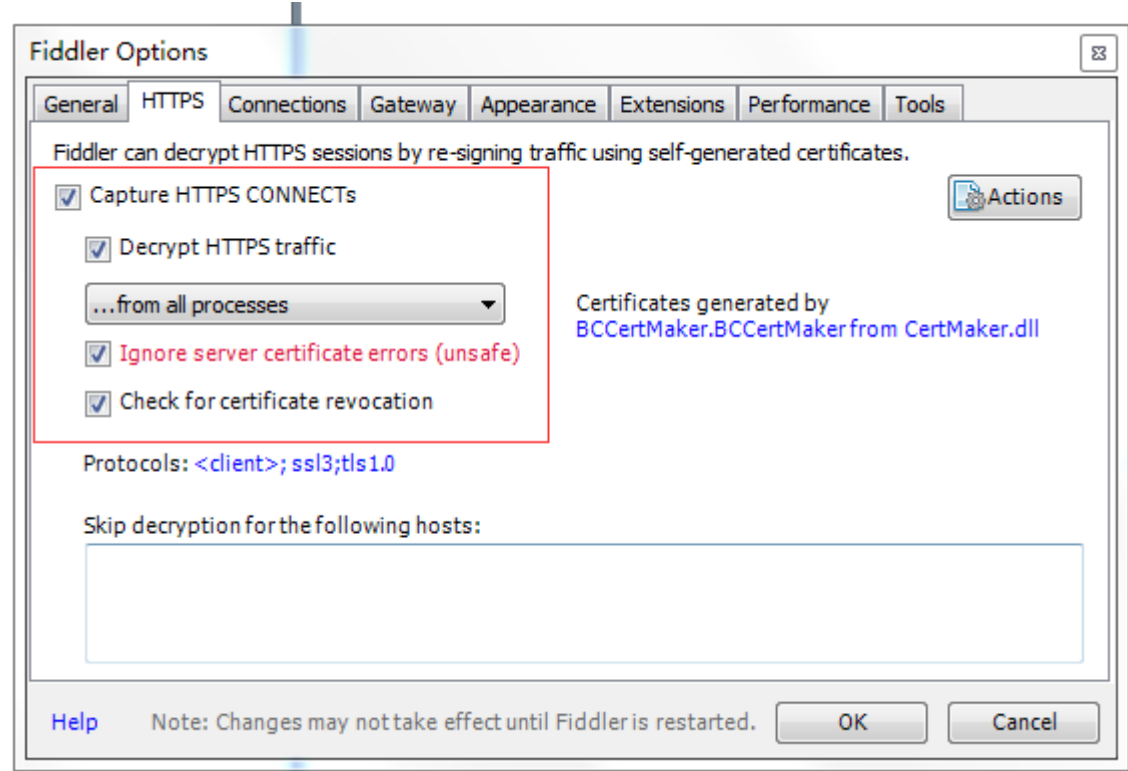


```
C:\Windows\System32>pip install -r D:\releate_package.txt
Requirement already satisfied: requests in c:\python36\lib\site-packages (from -r D:\releate_package.txt (line 1))
Requirement already satisfied: nose in c:\python36\lib\site-packages (from -r D:\releate_package.txt (line 2))
Requirement already satisfied: nose_htmloutput in c:\python36\lib\site-packages (from -r D:\releate_package.txt (line 3))
Requirement already satisfied: urllib3<1.23,>=1.21.1 in c:\python36\lib\site-packages (from requests->-r D:\releate_package.txt (line 1))
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\python36\lib\site-packages (from requests->-r D:\releate_package.txt (line 1))
Requirement already satisfied: idna<2.6,>=2.5 in c:\python36\lib\site-packages (from requests->-r D:\releate_package.txt (line 1))
Requirement already satisfied: certifi>=2017.4.17 in c:\python36\lib\site-packages (from requests->-r D:\releate_package.txt (line 1))
Requirement already satisfied: Jinja2 in c:\python36\lib\site-packages (from nose_htmloutput->-r D:\releate_package.txt (line 3))
Requirement already satisfied: MarkupSafe>=0.23 in c:\python36\lib\site-packages (from Jinja2->nose_htmloutput->-r D:\releate_package.txt (line 3))
```

4.2: 抓包工具 Fiddler

Fiddler 安装不说了，一路下一步。

对移动端 APP 的 https 抓包的话还需要如下设置(Tools→Fiddler Options)



保证移动端和 PC 端的网络在同一个局域网内，设置移动端代理为 PC 端的 ip 地址，端口号是 8888，然后下载并安装 Fiddler 根证书。





设置完成后，我这里以进入钉钉里的微应用为例演示下：





可以看到 Fiddler 中截取的报文

The screenshot shows the Fiddler interface. On the left, there's a list of intercepted requests with columns for Result, Protocol, Host, and URL. The selected request is a POST to 'https://testdingtalkapi3.xbongbong.com/dingtalk/auth/config.do'. The right pane shows the details of this request, including headers, body, and cookies. The body is a JSON object containing authentication information like 'agentId', 'appId', 'corpId', 'jsticket', 'nonceStr', 'signature', and 'timeStamp'. The status bar at the bottom indicates 'msg=操作成功' (msg=Operation successful).

五：框架设计

5.1、总体框架概览




```

22
23 # 接口用例中用到的时间格式
24 # 当前分钟
25 now = arrow.now().format('YYYY-MM-DD HH:mm')
26 # 下一小时
27 next_hour = arrow.now().shift(hours=1).format('YYYY-MM-DD HH:mm')
28 # 当天
29 today = arrow.now().format('YYYY-MM-DD')
30 # 明天
31 tomorrow = arrow.now().shift(days=1).format('YYYY-MM-DD')
32
33 # 用例数据文件
34 project_dir = os.path.abspath(os.path.dirname(__file__))
35 case_file = str(project_dir[0:-5]) + str("CaseData\\api_case_set.json")
36
37
38 # 生成随机签名
39 def random_sign(request_params, access_token):
40     parameters = str(str(request_params) + access_token).encode('utf-8')
41     return hashlib.sha256(parameters).hexdigest()
42
43
44 # 根据用例名获得用例参数
45 def get_case_parameters(api_name, case_name):
46     with open(case_file, 'r', encoding='utf-8') as file_path:
47         element_json = json.load(file_path)
48         # 暂存用例参数
49         initial_parameters = element_json[api_name][case_name]['parameters']
50         parameter_keys = initial_parameters.keys()
51         # 带 defaultData 关键字的参数类型
52         if 'defaultData' in parameter_keys:
53             default_data = initial_parameters['defaultData']
54             default_data_keys = default_data.keys()
55             # 循环 defaultData 的 key 值
56             for inner_key in default_data_keys:
57                 inner_value = default_data[inner_key]
58                 # defaultData 中 key 对应的 value 字符串中有 @ 符号 即：依赖已有数据 需要替换掉参数

```

1: Basic 包中封装请求发送, hash 加密, 读取用例数据, 断言, 并且所有的常量均写在此处。

2: CaseData 目录下放置的是测试用例数据, 以 Json 文件统一保存维护接口用例数据。

3: TestSuites 为了管理接口用例执行顺序且方便执行出错后问题定位, 此处放置的接口用例。

4: Report 此处放置的是 html 格式的测试报告。

5.2、庖丁解牛

5.2.1、用例数据管理

为了移至性考虑, 我这边以 Json 文件来保存用例数据, 同学可根据自己的情况判断使用数据库, excel 等等。

用例形式有两种:

1: 一个接口有多个用例, 也即: 多个用例共用一个 Url, 形式如下:



```

1  {
2  "create_data":{
3  "create_customer":{"id": 387457...},
61 "create_connector":{"id": 3792...},
114 "create_follow_record":{"id": 10178...},
145 "create_communicate_plan":{"url": "/communicatePlanApi/save.html"...},
170 "create_opportunity":{"url": "/opportunityApi/save.html"...},
232 "create_order":{"id": 5354...},
297 "create_supplier":{"url": "/supplier/save.html"...},
367 "create_purchase_order":{"url": "/purchase/save.html"...},
431 "create_in_stock":{"id": 866...},
491 "create_returned_purchase":{"url": "/returnedPurchase/save.html"...},
562 "create_receive_fee_plan":{"url": "/paymentApi/save.html"...},
588 "create_receive_fee_sheet":{"id": 1611...},
615 "create_pay_plan":{"url": "/payPlan/save.html"...},
646 "create_pay_sheet":{"id": 428...},
673 "create_assemble":{"id": 221...},
718 "create_out_stock":{"id": 789...},
761 "create_inventory":{"url": "/inventory/save.html"...},
807 "create_transfer":{"url": "/transfer/save.html"...}
850 },
851 "save_customer":{
852 "create_customer_with_exist_name":{...},
891 "create_customer_without_key_defaultData":{...},
930 "create_customer_without_key_name":{...},
969 "create_customer_without_name_value":{...},
1008 "create_customer_with_invalid_birthday":{...},
1047 "create_customer_without_key_corpid":{...},
1086 "create_customer_without_corpid_value":{...},
1125 "create_customer_without_key_nowuserid":{...},
1164 "create_customer_without_nowuserid_value":{...},
1203 "create_customer_with_other_corp_info":{...},
1242 "url":"/customerApi/save.html"
1243 },

```

2: 每个用例对应的 Url 均不同，我这里是把所有创建测试数据的用例写在了 create_data 中

```

1  {
2  "create_data":{
3  "create_customer":{"id": 387457...},
61 "create_connector":{"id": 3792...},
114 "create_follow_record":{"id": 10178...},
145 "create_communicate_plan":{"url": "/communicatePlanApi/save.html"...},
170 "create_opportunity":{"url": "/opportunityApi/save.html"...},
232 "create_order":{"id": 5354...},
297 "create_supplier":{"url": "/supplier/save.html"...},
367 "create_purchase_order":{"url": "/purchase/save.html"...},
431 "create_in_stock":{"id": 866...},
491 "create_returned_purchase":{"url": "/returnedPurchase/save.html"...},
562 "create_receive_fee_plan":{"url": "/paymentApi/save.html"...},
588 "create_receive_fee_sheet":{"id": 1611...},
615 "create_pay_plan":{"url": "/payPlan/save.html"...},
646 "create_pay_sheet":{"id": 428...},
673 "create_assemble":{"id": 221...},
718 "create_out_stock":{"id": 789...},
761 "create_inventory":{"url": "/inventory/save.html"...},
807 "create_transfer":{
808 "parameters":{"opUUID": "7E24bbeb0ed04cdcb0b7ceaa7ab096d9"...},
843 "url":"/transfer/save.html",
844 "transferId":236,
845 "expected_result":{
846 "code":1,
847 "msg":"成功"
848 }
849 }
850 },

```



每个用例的执行包括 3 部分:

1: 请求参数, 对应每个用例中 parameters 部分。

2: 接口地址, 对应每个用例中 Url 部分。

3: 预期结果, 对应每个用例中 expected_result 部分。

4: 最重要的部分, 如果需要依赖已有的数据, 此时请求参数需要替换, 形式是用例名称@需依赖的用例数据对应的 key

举个栗子:

```
"back_to_public":{
  "set_customer_to_public_pool":{
    "parameters":{
      "rows":[
        "create_customer@id"
      ],
      "corpid":"ding25c853585ddf93a335c2f4657eb6378f",
      "nowUserId":"030917160122954929"
    },
    "expected_result":{
      "code":1,
      "expected_data":"操作成功"
    }
  },
  "url":"/customerApi/setPublic.html"
}
```

这个用例是将用例退回公海池, 需要依赖 crete_customer 用例中的 id (即创建客户称后返回的客户 id)

5.2.2 基础操作封装

下面贴出来我这边封装的基础操作关键部分。

1: 请求中的加密签名

```
# 生成随机签名
```



```
def random_sign(request_params, access_token):
    parameters = str(str(request_params) + access_token).encode('utf-8')
    return hashlib.sha256(parameters).hexdigest()
```

2: 根据用例名称获得预期结果

根据用例名获得预期结果

```
def get_expected_result(api_name, case_name):
    with open(case_file, 'r', encoding='utf-8') as file_path:
        element_json = json.load(file_path)
        expected_content = element_json[api_name][case_name]['expected_result']
    return expected_content
```

3: 封装断言

软断言 即: 断言失败后继续执行

```
def soft_assertion_equal(expected_content, actual_content):
    # 判断 case 断言结果为 pass
    with soft_assertions():
        assert_that(expected_content).is_equal_to(actual_content)
```

软断言 包含

```
def soft_assertion_contains(expected_content, actual_content):
    # 判断 case 断言结果为 pass
    with soft_assertions():
        assert_that(str(actual_content)).contains(str(expected_content))
```

4: 封装 https 协议的 post 请求(请跟进自己实际情况封装)

封装 post 请求

```
def send_post(api_name, case_name):
    session = requests.session()
    # allow_redirects=False 获得当前请求(而不是跳转后那个请求)的数据
    cookies = session.get(host_for_test + login_info, headers=headers, allow_redirects=False).cookies
    # access_token 值
    token = cookies["xbbAccessToken"]
    # 获得测试用例的参数
    parameters = get_case_parameters(api_name, case_name)
    # 获得接口的 url 地址 创建数据和销毁数据时 必须指定接口名和用例名
    if ('create_data' == str(api_name)) or ('clear_data' == str(api_name)):
        url = get_url_of_data_case(api_name, case_name)
    # 非 set_up 和 tear_down 部分的用例 多个用例公用一个 url,所以 url 抽取为 api 的 key
    else:
        url = get_api_url(api_name)
    # 获得测试用例预期结果
```



```

sign_code = random_sign(parameters, token)
request_data = {"params": parameters, "sign": sign_code, "platform": "web", "frontDev": "0"}
send_request = session.post(url=host_for_test + url, data=request_data, headers=headers)
response_text = send_request.text
# 断言状态码和预期返回报文
expected_result = get_expected_result(api_name, case_name)
expected_code = expected_result['code']
expected_msg_text = str(expected_result['msg'])
with soft_assertions():
    assert_that(json.loads(response_text)).has_code(expected_code)
    assert_that(str(response_text)).contains(expected_msg_text)

```

5: 获得用例的请求参数(用例数据管理中需替换部分, 即用例名称@需依赖数据的 key)

根据用例名获得用例参数

```

def get_case_parameters(api_name, case_name):
with open(case_file, 'r', encoding='utf-8') as file_path:
    element_json = json.load(file_path)
    # 暂存用例参数
    initial_parameters = element_json[api_name][case_name]['parameters']
    parameter_keys = initial_parameters.keys()
    # 带 defaultData 关键字的参数类型
    if 'defaultData' in parameter_keys:
        default_data = initial_parameters['defaultData']
        default_data_keys = default_data.keys()
        # 循环 defaultData 的 key 值
        for inner_key in default_data_keys:
            inner_value = default_data[inner_key]
            # defaultData 中 key 对应的 value 字符串中有 @ 符号 即: 依赖已有数据 需要替换掉参数
            if '@' in str(inner_value):
                # defaultData 中 key 对应的 value 有 @ 符号, 且 value 类型是数组 []
                if isinstance(inner_value, list):
                    # 根据索引循环数组 找到 [] 数组中需要替换的值
                    for index in range(len(inner_value)):
                        final_param = inner_value[index]
                        # defaultData 中 key 对应的 value 有 @ 符号, 且 value 形式是数组 [], 并且数组中需替换的参数形式是 json 格式 {}
                        if isinstance(final_param, dict) and '@' in str(final_param):
                            # 循环数组里 json 对象中的 key
                            for final_key in final_param.keys():
                                if '@' in str(final_param[final_key]):
                                    case_name = str(final_param[final_key]).split('@')[0]
                                    value_id = str(final_param[final_key]).split('@')[1]
                                    # 判断需要替换的参数是否是时间

```



```

        if case_name == 'time':
            if value_id == 'today':
initial_parameters['defaultData'][inner_key][index][final_key] = today
                elif value_id == 'next_hour':
initial_parameters['defaultData'][inner_key][index][final_key] = next_hour
                elif value_id == 'now':
initial_parameters['defaultData'][inner_key][index][final_key] = now
                elif value_id == 'tomorrow':
initial_parameters['defaultData'][inner_key][index][final_key] = tomorrow
                # 需要替换的参数不是时间格式
            else:
                # 替换最终参数

        initial_parameters['defaultData'][inner_key][index][final_key] =
element_json['create_data'][case_name][value_id]
            else:
                continue
                # defaultData 中 key 对应的 value 有@符号,且 value 形式是数组,并且
                # 数组中需要替换的参数形式不是 json 格式
            elif '@' in str(final_param) and isinstance(final_param, dict) is False:
                case_name = str(final_param).split('@')[0]
                value_id = str(final_param).split('@')[1]
                # 判断需要替换的参数是否是时间
                if case_name == 'time':
                    if value_id == 'today':
                        initial_parameters['defaultData'][inner_key][index] = today
                    elif value_id == 'next_hour':
                        initial_parameters['defaultData'][inner_key][index] =
next_hour
                    elif value_id == 'now':
                        initial_parameters['defaultData'][inner_key][index] = now
                    elif value_id == 'tomorrow':
                        initial_parameters['defaultData'][inner_key][index] =
tomorrow
                # 需要替换的参数不是时间格式
            else:
                initial_parameters['defaultData'][inner_key][index] =
element_json['create_data'][case_name][value_id]
            else:
                continue
                # defaultData 中 key 对应的 value 有@符号,且 value 类型是 json
            elif isinstance(inner_value, dict):
                # 循环 json 对象中的 key
                for key in inner_value.keys():

```



```

        if '@' in inner_value[key]:
            case_name = str(inner_value[key]).split('@')[0]
            value_id = str(inner_value[key]).split('@')[1]
            # 判断需要替换的参数是否是时间
            if case_name == 'time':
                if value_id == 'today':
                    initial_parameters['defaultData'][inner_key][key] = today
                elif value_id == 'next_hour':
                    initial_parameters['defaultData'][inner_key][key] =
next_hour

                elif value_id == 'now':
                    initial_parameters['defaultData'][inner_key][key] = now
                elif value_id == 'tomorrow':
                    initial_parameters['defaultData'][inner_key][key] =
tomorrow

            # 需要替换的参数不是时间格式
            else:
                initial_parameters['defaultData'][inner_key][key] =
element_json['create_data'][case_name][value_id]
                # defaultData 中 key 对应的 value 有@符号,且 value 形式不是数组也不是
json, 而是普通字符串
            else:
                case_name = str(inner_value).split('@')[0]
                value_id = str(inner_value).split('@')[1]
                # 判断需要替换的参数是否是时间
                if case_name == 'time':
                    if value_id == 'today':
                        initial_parameters['defaultData'][inner_key] = today
                    elif value_id == 'next_hour':
                        initial_parameters['defaultData'][inner_key] = next_hour
                    elif value_id == 'now':
                        initial_parameters['defaultData'][inner_key] = now
                    elif value_id == 'tomorrow':
                        initial_parameters['defaultData'][inner_key] = tomorrow
                # 需要替换的参数不是时间格式
                else:
                    initial_parameters['defaultData'][inner_key] =
element_json['create_data'][case_name][value_id]
                    else:
                        continue
                # 请求参数中没有 defaultData 关键字
            else:
                # 循环请求参数,判断是否有需要替换的值(即:是否包含@字符)
                for key in parameter_keys:
                    value = initial_parameters[key]

```




```

if '@' in str(value):
    # 请求参数没有 defaultData,且需要替换的参数是数组类型
    if isinstance(value, list):
        # add-date: 2017-11-12 数组格式的值保存错误
        case_name = str(value[0]).split('@')[0]
        value_id = str(value[0]).split('@')[1]
        # 判断需要替换的参数是否是时间
        if case_name == 'time':
            if value_id == 'today':
                initial_parameters[key] = today
            elif value_id == 'next_hour':
                initial_parameters[key] = next_hour
            elif value_id == 'now':
                initial_parameters[key] = now
            elif value_id == 'tomorrow':
                initial_parameters[key] = tomorrow
        # 需要替换的参数不是时间格式
        else:
            value_need_to_transfer =
[element_json['create_data'][case_name][value_id]]
            initial_parameters[key] = value_need_to_transfer
    else:
        case_name = str(value).split('@')[0]
        value_id = str(value).split('@')[1]
        # 判断需要替换的参数是否是时间
        if case_name == 'time':
            if value_id == 'today':
                initial_parameters[key] = today
            elif value_id == 'next_hour':
                initial_parameters[key] = next_hour
            elif value_id == 'now':
                initial_parameters[key] = now
            elif value_id == 'tomorrow':
                initial_parameters[key] = tomorrow
        else:
            initial_parameters[key] =
element_json['create_data'][case_name][value_id]
    else:
        continue
# 用例参数的 key 值
return str(initial_parameters)

```

5.2.3、集中管理接口用例

TestSuite 中集中管理接口用例



```

1149
1150
1151 # 删除付款计划
1152 def test282_delete_pay_plan():
1153     request.send_post('clear_data', 'delete_pay_plan')
1154
1155
1156 # 删除回款单
1157 def test283_delete_receive_fee_sheet():
1158     request.send_post('clear_data', 'delete_receive_fee_plan')
1159
1160
1161 # 删除回款计划
1162 def test284_delete_receive_fee_plan():
1163     request.send_post('clear_data', 'delete_receive_fee_plan')
1164
1165
1166 # 删除采购出库单
1167 def test285_delete_returned_purchase():
1168     request.send_post('clear_data', 'delete_returned_purchase')
1169
1170
1171 # 删除采购合同
1172 def test286_delete_purchase():
1173     request.send_post('clear_data', 'delete_purchase')
1174
1175
1176 # 删除供应商
1177 def test287_delete_supplier():
1178     request.send_post('clear_data', 'delete_supplier')
1179
1180
1181 # 删除合同
1182 def test288_delete_order():
1183     request.send_post('clear_data', 'delete_order')

```

5.2.4、用例执行

用例执行可以使用 nosetests 命令来，当然可以在 Jenkins 中配置 job 有提交触发。

```

nosetests E:\XbbAPI\TestSuite\test_ApiSuites.py
--with-html --html-file=E:\XbbAPI\Report\report.html

```

解释如下：

nosetests E:\CRMAuto\TestSuites\test_customer_manage.py(执行那些 case，参数可以是具体的 py 文件或者目录) --with-html(开启 html 报告) --html-file=E:\CRMAuto\ReportAndLog\customer_report.html(指定 html 报告文件)



```

    assert_that(str(json.loads(response_text)[result_key])).contains(expected_t
st_data)
File "c:\python36\lib\contextlib.py", line 88, in __exit__
    next(self.gen)
File "c:\python36\lib\site-packages\assertpy\assertpy.py", line 71, in soft_as
sertions
    raise AssertionError(out)
AssertionError: soft assertion failures:
1. Expected <201005> to be equal to <1>, but was not.
----- >> begin captured logging << -----
urllib3.connectionpool: DEBUG: Starting new HTTPS connection (1): testdingtalk3.
xbongbong.com
urllib3.connectionpool: DEBUG: https://testdingtalk3.xbongbong.com:443 "GET /use
r/autoLogin.do?t=63t42AXnCd6hCEdQl+d0ZCXVIiwZVkrhN/PTN1xwhsw715sagLWrDX5zepLwtR3
Z7igSfah+CNgNbYzCwW1d9Q==&nonce=ldx013 HTTP/1.1" 302 0
urllib3.connectionpool: DEBUG: https://testdingtalk3.xbongbong.com:443 "POST /cu
stomerApi/batchCustomerGrab.html HTTP/1.1" 200 135
----- >> end captured logging << -----

-----
Ran 289 tests in 131.274s
FAILED (errors=2, failures=2)
C:\Windows\System32>

```

5.2.5 测试报告

Report 目录下生成 html 格式的测试报告



Overview

Class	Fail	Error	Skip	Success	Total
TestSuite.test_ApiSuites	2	2	0	285	289
Total	2	2	0	285	289

Failure details

TestSuite.test_ApiSuites (2 failures, 2 errors)

test218_search_customer_using_last_connect_time_no_data: **builtins.AssertionError**

test221_search_customer_using_update_time_have_data: **builtins.TypeError**

test223_search_customer_using_add_time_have_data: **builtins.KeyError**

test260_grab_customer: **builtins.AssertionError**



六：用例设计

6.1 对单个接口

使用等价类，边界值方法。

以新建客户为例，请求参数如下：

```

{"defaultData":{"name":"我是谁","phone":[{"name":"工作","telNum":"111"}],"source":"","industry":"","type":"","birthday":{"isLunarBirthdayShown":0,"birthday":320860800},"isIndividual":"","scale":"","nameShort":"","userId":[{"id":"1","name":"张杰"}],"instruction":"","importantDegree":0,"website":"","address":"","redundantData":{"attr3":"","attr40":"","attr7":"","attr9":"","attr38":""},"opUUID":"baba02a1e77e4a4f9744e143e9f2ad2d","corpid":"1","nowUserId":"1"}
    
```

设计用例如下(部分用例)

- 1: 联系方式重复 {"code":100015,"msg":"联系方式重复，客户保存失败"}
- 2: 缺失 key defaultData {"code":100019,"msg":"不可为空,defaultData 不可为空"}
- 3: 整个参数形式为 list {"code":100015,"msg":"参数格式错误"}
- 4: defaultData 参数形式为 list {"code":100015,"msg":"参数格式错误"}
- 5: 缺失 key name {"code":100019,"msg":"不可为空,客户名称 1 不可为空"}
- 6: 缺失 name 的值 {"code":100019,"msg":"不可为空,客户名称 1 不可为空"}
- 7: phone 格式为 dict {"code":100015,"msg":"参数格式错误,phone 格式错误"}
- 8: 缺失 phone 的 key name {"code":100015,"msg":"参数格式错误,phone 格式错误"}
- 9: 生日不为数字(unix time) {"code":100015,"msg":"参数格式错误"}
- 10: 缺失 key corpid {"code":100002,"msg":"参数缺失,corpid 不能为空"}
- 11: 缺失 corpid 的值 {"code":100002,"msg":"参数缺失,corpid 不能为空"}
- 12: 缺失 key nowUserId {"code":100002,"msg":"参数缺失,nowUserId 不能为空"}
- 13: 缺失 nowUserId 的值 {"code":100002,"msg":"参数缺失,nowUserId 不能为空"}
- 14: 使用其他公司的 corpid {"code":100012,"msg":"登录验证过期，请重新登录"}
- 15: 使用其他公司的 nowUserId {"code":100012,"msg":"登录验证过期，请重新登



录"}

16: 创建客户成功 {"code":1,"msg":"成功"}

6.2 多接口依赖

以尽量多的覆盖业务流为目的。

下面以删除回款单为例：

删除回款单→前提是有回款计划→前提是有合同→前提是有客户。

Step1: 创建客户

```
"create_customer": {
  "parameters": {
    "defaultData": {
      "name": "接口测试-客户",
      "nameShort": "接口执行-客户",
      "phone": [{"name": "工作", "telNum": "1881****846"}],
      "genre": "2", "type": "2", "isIndividual": "2", "scale": "2", "industry":
"3", "importantDegree": 3,
      "birthday": {
        "isLunarBirthdayShown": 0,
        "birthday": 585158400
      },
      "country": "中国",
      "address": {"province": "河南省", "city": "周口市", "district": "太康县", "address": "大许
寨乡", "location": [114.817203, 34.000259]},
      "source": "渠道代理",
      "website": "", "instruction": "",
      "userId": [
        {
          "id": "030917160122****29",
          "name": "姜林斌"
        }
      ]
    },
  },
}
```



```

    "redundantData": {"attr1": [], "attr2": [], "attr3": ""},
    "opUUID": "2b40b2a47008428c88e59dd38a1333cb",
    "corpid": "ding25c853585ddf93a335c2f4657eb6378f",
    "nowUserId": "030917160122954929"
  },
  "url": "/customerApi/save.html",
  "id": 387483,
  "expected_result": {
    "code": 1,
    "msg": "成功"
  }
}

```

Step2: 创建合同

创建合同时依赖 step1 中创建客户时返回的客户 id。

```

"create_order":{
  "parameters":{
    "defaultData":{
      "name":"接口测试-合同",
      "contractNo":"SO.20171119001",
      "customerName":{
        "id":"create_customer@id",
        "value":"接口测试-客户"
      },
    },
    "products":{
      "productList":[ {
        "id":40791,
        "name":"迈巴赫 exelero",
        "unit":"千克",
        "num":1,
        "price":9000000,
        "productPrice":9000000
      }
    ]
  }
}

```



```

    ],
    "discount":100,
    "otherCharge":0
  },
  "totalMoney":37000000,
  "signTime":"time@today",
  "status":"1",
  "type":"1",
  "signPerson":[
    {
      "id":"030917160122954929",
      "name":"姜林斌"
    }
  ],
  "payMethod":"1",
  "deadline":"time@today",
  "payment":""
},
"redundantData":{"opUUID":"08f9de200c614c00ba962f087d4679d9",
"corpid":"ding25c853585ddf93a335c2f4657eb6378f","nowUserId":"030917160122954929"
},
"url":"/contractApi/save.html",
"id":5376,
"expected_result":{
  "code":1,
  "msg":"成功"
}
}

```

Step3: 创建回款计划

创建回款计划时需依赖 step2 中创建合同时返回的合同 id。

```

"create_receive_fee_plan":{
  "parameters":{

```




```

"defaultData":{
  "contractId":"create_order@id",
  "payment":[
    {
      "amount":10000.01,
      "paymentNo":"接口-回款计划",
      "paymentType":"4",
      "estimateTime":"time@today",
      "status":"1",
      "memo":""
    }
  ],
  "opUUID":1511076696659
},
"corpid":"ding25c853585ddf93a335c2f4657eb6378f",
"nowUserId":"030917160122954929"
},
"url":"/paymentApi/save.html",
"ids":3580,
"expected_result":{
  "code":1,
  "msg":"成功"
}
}

```

Step4: 创建回款单

创建回款单需依赖回款计划 step3 中回款计划的 id

```

"create_receive_fee_sheet":{
  "parameters":{
    "defaultData":{
      "paymentSheetNo":"接口-回款单",
      "belongId":[
        {

```



```

        "id":"0933106141869064955",
        "name":"测试账号"
    }
],
"amount":999.99,
"paymentTime":"time@today",
"estimateTime":"time@today",
"payMethod":"1",
"memo":"",
"paymentId":"create_receive_fee_plan@ids"
},
"corpid":"ding25c853585ddf93a335c2f4657eb6378f",
"nowUserId":"030917160122954929"
},
"url":"/paymentSheetApi/save.html",
"id":1637,
"expected_result":{
    "code":1,
    "msg":"成功"
}
}

```

Step5: 删除回款单

删除已有的回款单需依赖 step4 中创建回款单时返回的回款单 id

```

"delete_pay_sheet":{
    "parameters":{
        "id":[
            "create_pay_sheet@id"
        ],
        "corpid":"ding25c853585ddf93a335c2f4657eb6378f",
        "nowUserId":"030917160122954929"
    },
    "url":"/payPlan/deletePaySheet.html",

```



```
"expected_result":{  
  "code":1,  
  "msg":"成功"  
}
```

七：持续集成

持续集成的就不多说啥了(之前写过一篇文章专门介绍使用 Jenkins 作为持续集成平台的文章,《Jenkins 持续集成实践》), 多多关注 51Testing 的《51 测试天地》电子期刊吧~



Python 调用安卓 adb 命令(下篇)

◆ 作者 : lamecho 辣么丑

1.1、概要

今天是《Python 自动化测试应用》的第十一篇，本篇将继续第十篇的 adb 命令结合 pyapp 框架编写过程中实际遇到的那些坑。后面的内容不是简单的列举命令（因为网络上大把，大家关心什么可以自己搜索），而是延续一贯的风格以实战为主。好了各位同学坐稳了，我要发车了，哈哈。

1.2、那些需要注意的 adb 命令

1.2.1、“adb shell input”

这条命令，在测试过程中也是经常用到的，它后面可以跟 tap, text, swipe, 进行点击屏幕，输入文本，滑屏的操作，具体在 python 中使用按照命令格式执行也不会出现什么问题。但是如果我们要长按某个元素，从而实现具体功能呢？下面给出实现代码。

```
def long_press(dev,data,hold_time):
    action='adb -s '+dev+' shell input touchscreen swipe '+'%d'%data[0][0]+' '+'%d'%(data[0][1])+' '+'%d'%data[-1][0]+' '+'%d'%(data[-1][1])+' '+hold_time
    print action
    pi= subprocess.Popen(action,shell=True,stdout=subprocess.PIPE)
    long_press('4d0041b1be98b01f',[[540,716],[545,718]],'1000')
```

可以看到我定义的 long_press 方法，action 里用到的仍然是 swipe 命令，大家知道 swipe 是滑动屏幕的操作，那么如果我们在传递滑动范围坐标的时候，设定的滑动范围非常小，那么是不是就是间接的达到了长按某一个区域的目的，然后配合一个整个命令的执行时间的参数，是不是就完美解决了长按这个动作。

我们看一看完整的 adb 命令：

```
adb -s 4d0041b1be98b01f shell input touchscreen swipe 540 716 545 718 1000
```



解释: -s 后跟设备号, swipe 先传移动坐标范围“540 716 545 718”, 然后 1000 是长按的时间, 1000 的单位是毫秒(注意)。

大家不要以为这条命令就这样介绍结束了, 还没有。按道理来说 input 后面的 source 源在 swipe 命令里默认就是 touchscreen, 所以一般我们在写这条命令时是可以省略 touchscreen 的, 但是实际我在编写 pyapp 框架的时候(由于 pyapp 设计需要支持多台设备), 拿了不同型号品牌(分辨率)的手机做适配, 毕竟框架要照顾的范围要必须广要有一定的普适性。我发现, 在 pyapp 的僚机模式下(就是一台主设备操控多台附属设备, 获取最新 pyswat 和 pyapp 程序), 主设备的长按操作在有些副设备上实现不成功, 开始我也是在命令里省略了 touchscreen。开始也想不到什么原因, 因为命令来说也不复杂不存在写错的情况, 毕竟有些副设备还是能正确响应动作的, 后来我把 shell input 命令的帮助结果打印出来研究了一下(见上一篇文章中有截图), 在 swipe 命令的参数传递中说明了 touchscreen 是缺省, 默认属性, 所以正常来说我们是不用在命令中明确指定该参数值的, 然而我也是瞎猫碰到死耗子, 试着在命令里加入了 touchscreen, 最后执行结果每一台副设备在长按命令的执行上都成功了。

1.2.2、adb 命令如何输入中文?

adb 命令里进行输入文本输入 ‘adb -s 设备号 shell input text 输入的内容’。在原生的 adb 命令里是不支持中文输入的, 所以我们在测试的时候只能输入英文字符。然而实际我们在做 app 测试的时候避免不了需要输入中文字符的情况, 这里给大家介绍一种曲线救国的办法。利用“ADBKeyBoard”输入法来进行中文的输入, 通过广播的方式达到输入中文字符, 具体命令: adb shell am broadcast -a ADB_INPUT_TEXT --es msg “内容”。

1.2.3、启动应用

在 appium 的应用中每次只能开启一个 app, 而如果用 adb 命令的话就灵活许多, 输入: adb shell am start -n package 名/.activity 名, 这里的 package 名和 activity 名和 appium 中配置的一致。比如我们启动计算器程序, 对应的命令就是“adb shell am start -n com.android.calculator2/.Calculator”。这里要再提醒大家一点, APP 的 package 名和 activity 名一定要找对特别是 activity 名, 具体大家可以参看我的《python 自动化测试应用-第 2 篇 (APP 测试) --Appium 初识篇》里边讲解了具体的查找方法。当然大家也是可以根据 adb 命令去进行查找, 在 pyapp 测试框架中我就是根据 adb 命令进行名称的查找启动应用的。这里我讲解一下思路, 利用命令 adb shell pm list packages -3, 将手机中



安装的第三方 app 列举出来，然后通过时间比对找到新安装的 app 从而确定名称。其实大家只要有了思路可以在百度中查找对应的命令即可。

1.2.4、只会发短信，那么查看短信呢？不会你就 out 了！

网上你可能找到如何发短信，打电话的相关 adb 命令的介绍，其实原理也就是 1.2.3 中介绍的，还是启动对应的应用程序来实现。比如发短息：`adb shell am start -a android.intent.action.SENDTO -d smsto:发送号码 --es sms_body 短信内容`。那么如何读取一条短信内容呢？

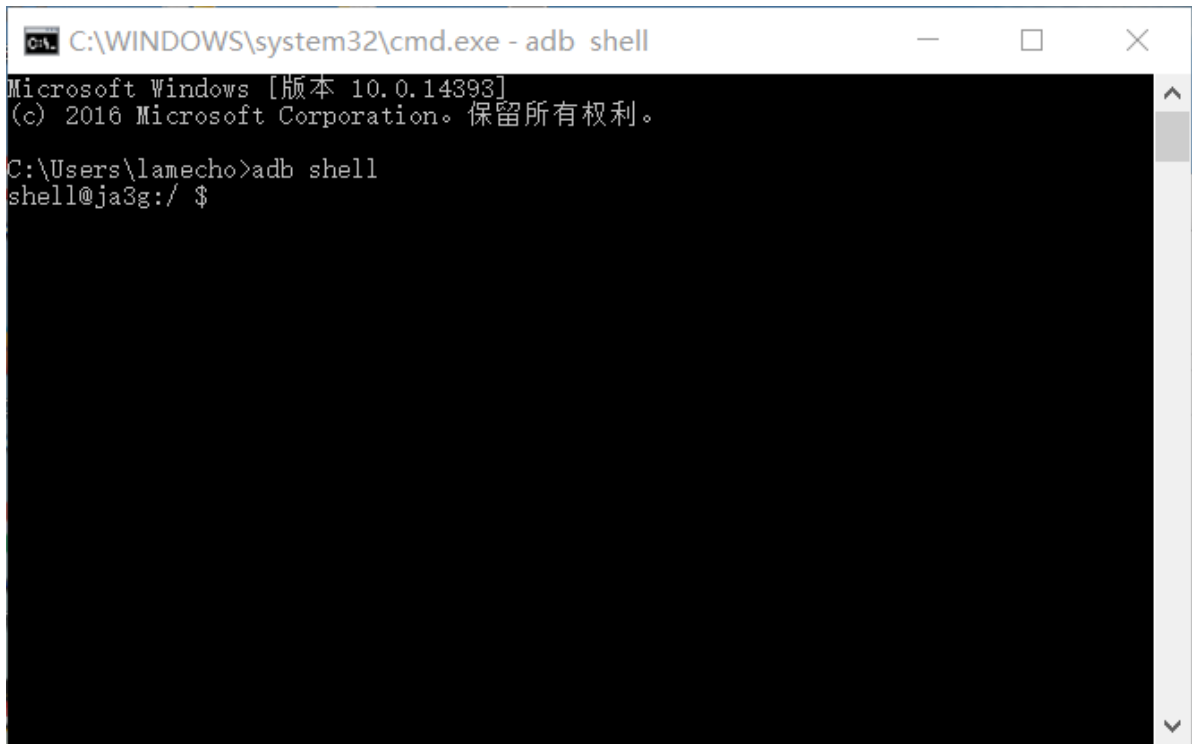
在编写 pyapp 框架的过程中为了实现 app 的验证码自动填写功能，着实费了一番功夫。目前大部分的注册登录都是可以用动态短信验证码来进行操作的，当然 app 开发者本身是可以实现在收到短信后读取验证码进行自动填写，遇到没有这种功能的 app 我们当然也可以按照此原理去实现。由于安卓手机的所有短信都是存储在数据库中，那么我们只要找到短信的这个数据库文件，自然就可以轻松的通过 python 的数据库操作读取到短信内容了。'/data/data/com.android.providers.telephony/databases/mmssms.db'这个路径下“mmssms.db”文件就是保存短信内容的数据库文件，那么剩下的工作自然就是数据库的读操作了，用正则表达式匹配到验证码即可，最后通过 adb 的 input 命令写入到 app 中即可。这里还需要注意的是，操作“mmssms.db”文件需要 root 权限，所以你要想在 pyapp 框架里使用这个功能必须是使用 root 过的手机设备。

1.2.5、adb 命令 su 权限如何使用？

什么是 adb 命令的 su 权限，举个简单的例子，比如说你要访问手机的内存 /data/system 路径下的文件。在 cmd 命令窗中你需要执行三步：

(1) 在 cmd 窗体中输入命令：`adb shell` 回车

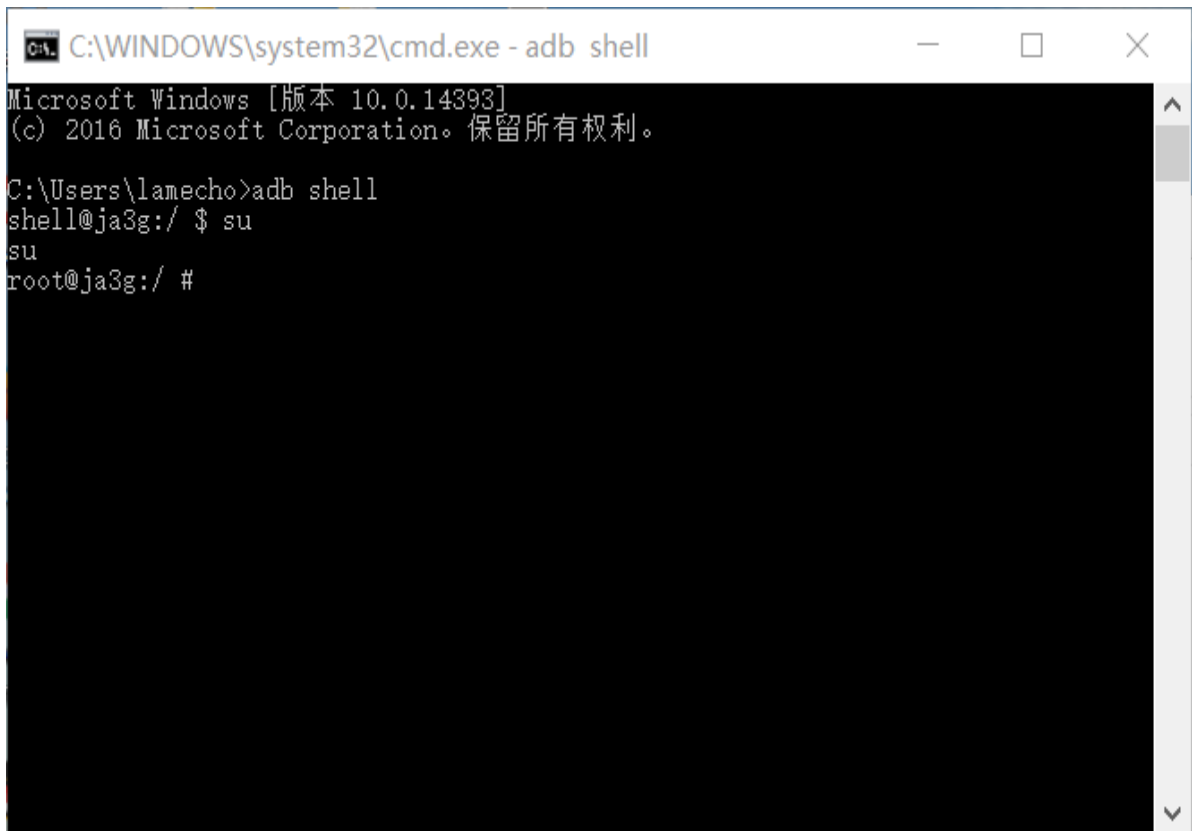




```
C:\WINDOWS\system32\cmd.exe - adb shell
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

C:\Users\lamecho>adb shell
shell@ja3g:/ $
```

(2) 输入 su 回车



```
C:\WINDOWS\system32\cmd.exe - adb shell
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

C:\Users\lamecho>adb shell
shell@ja3g:/ $ su
su
root@ja3g:/ #
```

可以看到命令符由\$变成了#符号。

(3) 输入 cd /data/system




```

C:\WINDOWS\system32\cmd.exe - adb shell
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Users\lamecho>adb shell
shell@ja3g:/ $ su
su
root@ja3g:/ # cd /data/system
cd /data/system
root@ja3g:/data/system #
    
```

这样就完成了/data/system 路径的访问，如果遇到那些需要权限的文件你没有执行 su 的话，可能就会给你返回一句 Permission denied（权限拒绝）。

好了，说了这么多上面都是在 cmd 窗口中去分步执行的，那么在 python 中怎么去实现呢？如果按照之前讲解的方式在 python 脚本中分别执行这三条命令，肯定是不行的。因为程序执行一次 adb 命令，就会建立一个独立的进程，所以我们在脚本中不能按顺序执行三次 adb 命令，这样是达不到效果的。我们必须让一条 adb 命令一次完成所有的步骤的执行才对，那么我们这条命令该怎么写呢？简单暴力点，像这样：adb shell su cd data\system

肯定是不行的，不过大致意思是对的，只是具体写法上要按照正确的格式在 su 后面加上 -c 就行了。如：adb shell su -c cd "data\system"。我们看看这条命令，注意以后需要 su 权限的 adb 命令都可以这样写，-c 后面跟着具体的操作命令即可。

1.2.6、uiautomatorviewer 和 hierarchyviewer 傻傻分不清楚？

大家在做 android 自动化测试时，必定会需要知道界面元素/控件的相关属性，如 id, class 等，这时在 androidsdk 的 tool 工具中就会用到 uiautomatorviewer 和 hierarchyviewer。这两个工具都可以很直观的方便大家去查找界面元素，而这样只是单



纯的人工借助工具去查看，既然是要做自动化测试，必然我们要去通过代码编程去代替手工操作了。在 pyapp 框架中，我也是利用 uiautomatorviewer 来捕获 app 界面的从而获取对应的元素/控件。接下来我们就来看看 python 是如何做到的。相对来说 uiautomatorviewer 的实现更容易些，一条 adb 命令就可以了。

```
order='adb -s device shell uiautomator dump'
```

运行成功后会返回：

```
“UI hierchary dumped to: /storage/emulated/legacy/window_dump.xml “
```

结果很直观了，这个 window_dump.xml 里就是整个界面的布局层级信息，从中我们就可以获取到各个元素/控件的属性信息。如果利用浏览器打开 xml 文件大家也可以直观的看到界面的层级关系，如果大家只关心具体的元素控件，我们可以在命令后面加上—compressed，这样获取的 xml 就会清爽许多。

我们接着看看怎么在 python 中利用 hierarchyviewer 实现元素/控件获取，首先我们要知道要实现 hierarchyviewer 的元素获取我们要打开手机的 View Server 服务，并与其进行 socket 通信，从而获取到元素/控件信息。那么接下来我们将通过几条不同的 adb 命令来准备好与 View Server 进行通信的环境。

第一步：

通过 “adb shell service call window 3” 命令得到返回值 Result: Parcel(00000000 00000001 '.....')或者 Result: Parcel(00000000 00000000 '.....')如果是 00000001 表示 View Server 是开启的，反之我们就要开启 View Server 服务。

第二步：

如果 View Server 没有打开，我们就通过 “adb shell service call window 1 i32 4939” 命令打开。然后再通过第一步的命令确认是否开启了 View Server。

第三步：

当我们开启了服务后，需要再将手机的 4939 端口映射到电脑的 4939 端口，这样就可以进行 socket 通信了。命令为 “adb forward tcp:4939 tcp:4939”

以上这三步完成后，接下来我们就可以在 python 脚本中与 View Server 建立 socket 通信了。由于已经有服务端 View Server，我们只需要用 python 实现客户端的 socket 代



码即可。

```

import socket#导入 socket 模块
sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)#建立 TCP 连接
sock.connect(('127.0.0.1',4939))#连接服务端地址及端口并建立连接
sock.send('list')#发送 list 命令向服务端
sock.recv(1024)#接收返回结果
  
```

这里就不去讲解 python 的 socket 用法了，重点讲解一下这里的 list 指令。当我们建立好与 View Server 的通信后，就可以发送不同的指令向服务端，获取信息了，这里的 list 意思就是获取当前手机活动界面的信息。如下图

```

42f57bf0 com.android.systemui.ImageWallpaper
432b8608 com.miui.home/com.miui.home.launcher.Launcher
437fb410 InputMethod
42f46fc8 AppNotificationPanel
4326be40 KeyguardScrim
42e89ad0 SViewCover
43288b90 StatusBarBlur
43236408 StatusBar
433c7ed8 RecentsPanel
435ddc40 FloatNotificationPanel
43166ca0 SPenGesture
DONE.
  
```

有了这个信息，我们就可以继续发送 dump 命令去获取具体某个界面的详细层级信息了。举个例子比如我们要打印出上图中第二行

“432b8608 com.miui.home/com.miui.home.launcher.Launcher”

的界面信息，在我们发送内容中应该这样写 sock.send(“dump 432b8608”),获取结果如下图。由于内容会比较多，建议大家在代码中将返回值写入 txt 文件中，方便查看。具体的内容中每一段表示一个元素，里边详细的列举了元素的各种属性信息，如 id, class, 坐标等。



```
g:isOpaque()=5,false isSelected()=0,false isSoundEffectsEnabled()=4,true drawing:willNotCacheDrawing()=5,false drawing:willNotDraw()=5,false
    com.miui.home.launcher.ShortcutIcon@44f7d3c8 drawing:mForeground=4,null padding:mForegroundPaddingBottom=1,0 padding:mForegroundPaddingLeft=1,0
padding:mForegroundPaddingTop=1,0 drawing:mForegroundInPadding=4,true measurement:mMeasureAllChildren=5,false drawing:mForegroundGravity=3,119 events
events:mLastTouchDownX=3,0.0 events:mLastTouchDownIndex=2,-1 mGroupFlags=7,2244624 layout:mChildCountW
ithTransientState=1,0 focus:getDescendantFocusability()=24,FOCUS_BEFORE_DESCENDANTS drawing:getPersistentDrawingCache()=9,SCROLLING drawing:isAlwaysDr
drawing:isChildrenDrawingOrderEnabled()=5,false drawing:isChildrenDrawnWithCacheEnabled()=5,false paths:mXmlFilePath=4,null bg_=4,null paths:mBackgro
measurement:mMeasuredHeight=3,270 measurement:mMeasuredWidth=3,234 measurement:mMinHeight=1,0 measurement:mMinWidth=1,0 padding:mPaddingBottom=1,0 pa
padding:mPaddingTop=1,0 mPrivateFlags_DRAWN=4,0x20 mPrivateFlags=8,16812208 mID=5,NO_ID layout:mRight=3,276 scrolling:mScrollX=1,0 scrolling:mScrollY
mSystemUiVisibility=1,0 layout:mTop=3,318 layout:mBottom=3,588 padding:mUserPaddingBottom=1,0 padding:mUserPaddingEnd=11,-2147483648 padding:mUserPac
serPaddingStart=11,-2147483648 mViewFlags=9,404766849 drawing:getAlpha()=3,1.0 layout:getBaseline()=2,-1 accessibility:getContentDescription()=6,Root#
accessibility:getImportantForAccessibility()=3,yes accessibility:getLabelFor()=2,-1 layout:getLayoutDirection()=22,RESOLVED_DIRECTION_LTR layout_x=2,
layout:layout_endMargin=11,-2147483648 layout:layout_leftMargin=1,0 layout:layout_mMarginFlags_LEFT_MARGIN_UNDEFINED_MASK=3,0x4 layout:layout_mMargin
layout:layout_mMarginFlags=2,12 layout:layout_rightMargin=1,0 layout:layout_startMargin=11,-2147483648 layout:layout_topMargin=1,0 layout:layout_hei
drawing:getPivotY()=5,135.0 layout:getRawLayoutDirection()=7,INHERIT text:getRawTextAlignment()=7,GRAVITY text:getRawTextDirection()=7,INHERIT drawin
ng:getRotationY()=3,0.0 drawing:getScaleX()=3,1.0 drawing:getScaleY()=3,1.0 getScrollBarStyle()=14,INSIDE_OVERLAY drawing:getSolidColor()=1,0 getTag()
text:getTextDirection()=12,FIRST_STRONG drawing:getTranslationX()=3,0.0 drawing:getTranslationY()=3,0.0 getVisibility()=7,VISIBLE layout:getWidth()=3
=5,false layout:hasTransientState()=5,false isActivated()=5,false isClickable()=4,true drawing:isDrawingCacheEnabled()=5,false isEnabled()=4,true foc
focus:isFocused()=5,false isHapticFeedbackEnabled()=4,true isHovered()=5,false isInTouchMode()=4,true layout:isLayoutRtl()=5,false drawing:isOpaque()
drawing:willNotCacheDrawing()=5,false drawing:willNotDraw()=4,true
```

至此，我们了解到通过发送 `list`，`dump` 两个命令可以进行元素信息的获取。另外还有一个 `capture` 命令可以获取到元素在界面中的截图，这里就不再赘述了。至于到底是用 `uiautomatorviewer` 还是 `hierarchyviewer` 去实现元素的获取，决定权就在你的手上了。

好了，本次的内容就写到这里，本篇着重讲解了一些非常规的 `adb` 命令的使用，尽可能的贴住适合我们做自动化测试相关的 `adb` 命令使用，或者说是针对我在写 `pyapp` 时实现的一些比较实用的 `adb` 命令。最后再声明本文不去列举 `adb` 命令，网上资料有很多，最后希望本篇文章能够帮助到大家。



测试人员如何把控项目进度

◆ 作者：瑾沐沐

项目背景简介

项目代称	K 项目
项目成员	6 人（1 个测试猿+5 个程序猿）
项目周期	两个月（截止日期，国庆节前）
工时评估	以天为单位（模糊评估）

测试猿的窘境：

- 1、需求文档不明确？
- 2、提测时间不明确？
- 3、项目进度不明确？
- 4、我是谁？我该干嘛？

想必每个测试猿都会遇到以上的窘境，版本到项目快截止时才提测，最后项目延误了，又要默默的背锅？

项目进行了半个月，依然没有我什么事儿，我真的不想国庆加班啊，去年就已经安排了今年的国庆节行程，怎么可能延误，必须要改变现状了…

一、主动沟通，抛出问题

主动找研发经理沟通，抛出问题，提出解决方案；迈出这一步我也是三思而后行。

1、找到沟通有效的人

- 找项目负责人？项目负责人只是其中一个研发，解决不了根本的问题。



- 找项目经理？项目经理经常出差，很难得到准确的回复。
- 最后，只能冒昧约研发经理谈话了，程序猿的直属领导，应该是最有话语权了。

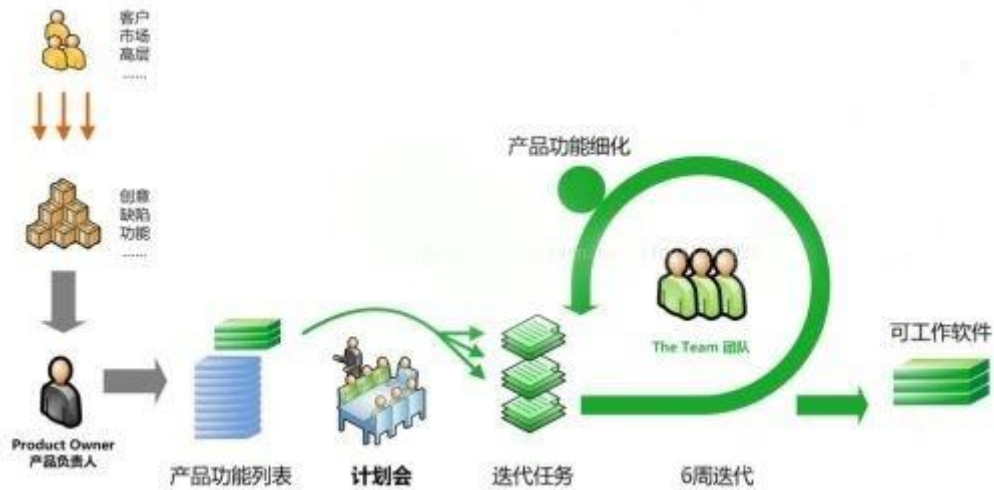
2、抛出问题，提出解决方案

存在问题：

- 需求不明确，没有相关文档输出
- 任务没有划分优先级
- 任务工期评估模糊
- 按目前的进度，国庆前不可能完成该项目

解决方案：

- 工时精准评估，以小时为单位
- 提供产品待办列表，输出任务优先级、研发进度、提测日期等信息
- 进行迭代开发，三期迭代，每个迭代为期两周(离项目截止日期正好6周)



沟通结果：

- 第一点被否定，可以尝试进行迭代开发
- 测试（me）主动提供迭代需求清单模板
- 测试提前介入，先行接口测试，后续功能测试



二、迭代开发，积极推进

- 1、我主动提供迭代开发需求管理模板（如下截图）；
- 2、总共三期迭代，每期迭代历时两周；
- 3、周一：项目负责人邮件发出一期迭代需求清单，抄送项目干系人；
- 4、周五：测试负责人，总结项目进度，邮件发送项目干系人；
- 5、每期迭代结束，总结本期迭代的完成率以及优缺点，要生成可交付的产品；

需求编号	模块	子模块	需求描述	优先级	开发状态	提测日期	测试状态	研发人员	测试人员
			需求描述：明确本期迭代任务						
			优先级：研发和测试明确工作重心						
			开发状态：明确任务进度						
			提测日期：给研发施加无形的压力，测试明确什么时候提测版本						
			测试状态：明确任务是否完成						

三、迭代结束，项目完结

经过为期 6 周的迭代开发，团队小伙伴的不懈努力、研发经理的不断施压；项目最终按时完结，回归测试也提前完成，终于可以安心庆国庆、过中秋喽 ...

四、测试经验总结

1、如何实现测试左移

- 需求阶段介入，明确需求甚至可以给出自己的对产品的设计意见
- 先行接口测试，尽早发现接口层面的问题，可避免后期测试浪费时间
- 重视数据库测试，新的项目所有的表都是新建的，可以从表结构、字段、索引等各个方面把关，遇到问题前期修改成本较低

2、多版本并行，如何高效执行测试任务

由于我一个测试猿要对接五个程序猿，某天出现了同时提测四个版本的情况，在片



刻的慌乱后我采取了以下方式：

- 提测任务按优先级排序，进行一轮主功能测试，使每个程序猿手头都有缺陷要处理；
- 对优先级较高的任务进行第二轮全功能覆盖测试
- 回归缺陷，之后再进行三轮冒烟测试，发现新的缺陷，绝对不能让研发空闲
- 就像玩游戏一样，轮番向各个研发扔 bug，直到所有 bug 关闭才 game over

3、迭代开发、敏捷测试

由于我大学毕业后就加入了敏捷开发团队，敏捷（scrum 模式）对我的影响很深，一直想在现在项目中推行敏捷，但是大家都不愿意拥抱变化，敏捷最大的一个特点就是“拥抱变化”。因此本次项目就采取了迭代开发的模式，用敏捷的思维进行测试

- 当面沟通，减少信息误差
- 持续改进，及时反馈问题
- 响应变化，停止推脱抱怨

4、有效管理缺陷，缩短项目进度

敏捷中注重处理 bug 的效率，发现问题快，修复起来也较快；我在实际的测试中采用了传统与敏捷结合的方式处理缺陷，减低了 bug 修复的成本，有效缩短了项目周期；具体总结如下：

- 页面缺陷，集中反馈，提供截图说明；
- 需求缺陷，双方沟通，达成共识后记录缺陷，可降低时间成本；
- 面对面沟通，主动重现解释 bug；
- 重视缺陷的成本，高时间成本，低价值的缺陷建议口头通知解决，低时间成本，高价值的缺陷需要记录追踪；
- bug 及时跟进：及时验证、及时反馈、及时关闭；

总而言之：主动沟通、当面沟通、耐心沟通、持续沟通……



(以下是项目过程中发生的段子)

测试猿: 嘿, 下午茶来了, 吃个香蕉休息一下吧

程序猿: 谢谢啊

测试猿: 有几个 bug, 现在给你重现一下吧

程序猿: 好啊

测试猿: 今天必须解决这几个问题哈

程序猿: 我很忙啊, 没空处理啊

测试猿: 别吃香蕉了, 立刻、马上处理……

程序猿: @#¥%&*^~#



如何做可达性测试？

◆ 作者：zjyforuok

什么是可达性(Accessibility)

可达性(Accessibility)有一个通俗易懂的等价名词叫“无障碍”，它体现在生活的各个方面，比如盲道，商场医院的无障碍通道，机场的无障碍电梯等都是“可达性”的一种实现。

具体到软件行业，可达性是指软件从设计到实现，能使残障人士，老年人，非本土人群等获得同等信息和服务。简言之，如果你的软件能够支持不同的人群来使用，那么你就做到了可达性。

举个例子，如今很多老年人也喜欢网上购物，但是手机上的字体和图片都很小，老人可能看不清楚。如果购物网站或者手机本身支持放大字体和图片，并在放大后保证足够的清晰度，就能使得老人购物也很方便。这就是可达性针对特定人群需要的一种实现。

欧美对软件可达性的要求比较高，比如美国有一个著名的 508 无障碍方案，1998 年 8 月 7 日由《劳动力投资法》(P.L.105-220)修改。强制要求在美售卖的软件要符合可达性，通过可达性测试并按政府要求的格式生成相应的可达性报告。

在国内，随着信息技术产业的快速发展，人们对软件可达性的需求也越来越高。

可达性的实现覆盖视觉、听觉以及认知等方面，下面我们来看看可达性测试所需要涵盖的基本测试点。

视觉和听觉相关的测试

颜色 / 图片配字 / 对比度

颜色相关的测试主要是针对色盲 / 色弱用户考虑的。我本人在几年前有过这样一个经历，一位同事讲 PPT，在接近绿色的背景上用了大红的字体。他用绿底色的本意是希



望让观众的眼睛比较舒服，而红色字体是希望引起大家的注意。但是有可达性测试经验的我就对此特别敏感，因为如果观看者里有红绿色盲，那这样的 PPT 对其几乎就是不可读的。

在软件开发的过程中，此类情况还是很常见的。比如柱状图，用不同的颜色做分类展示。多数人看到这样的图并不会觉得有什么不妥。但可达性测试人员对此要尤其注意，凡是单纯用颜色来区分或分类的实现都是典型的可达性 bug。

那么如何做才能给对颜色辨识有困难的用户提供同等的信息呢？最常用的一个做法是加文字标注。还拿柱状图做例子，比如用柱状图统计超市不同品牌的啤酒的售卖量，红色代表百威，绿色代表蓝带。那在颜色区分的同时，在红色柱旁加文字“百威”；在绿色柱旁边加文字“蓝带”。这样，即使不能分辨两种颜色，通过文字标注也可以获取等同的信息。

图片配字也是类似的原因和要求，少数人会对某些类型的图片识别有困难，或者无法辨识图片里不同颜色的差异，导致无法理解图片的内容。所以，如果软件的输出有非装饰性的图片——即希望通过图片向客户传达信息或指示的时候，需要配上相应的文字说明。

对比度则是指字体和其背景的颜色对比度 / 差异性要足够大（否则用户可能难以辨认文字内容），而且字体大小得满足特定需求。具体要求 W3C 有非常细节的定义，同时 W3C 也列举了一些工具帮助检验前景 / 背景的对对比度。内容较多，本文就不再赘述了。

标题 / 标签 / 感知觉描述

标题 / 标签主要是针对视觉障碍的人群。符合可达性软件的一个基本要求是要支持读屏工具，对于盲人或弱视人群，可以通过读屏工具来“读”软件，并且操作软件。因此软件里的标题 / 标签必须是有意义且描述准确的。这样的话，当读屏工具读到某个标题，使用者可以根据标题的内容来判断这部分内容是不是他 / 她感兴趣或需要的，从而决定是跳过还是进一步深入这个标题下面的具体内容。

在有些软件里我们会包含一些和感知觉相关的描述，例如形状，大小，位置，方位等等，对于大多数人群，这些描述为使用软件提供了便利性。但是一些有某类感知觉障碍的人可能无法理解其含义。



例如：“提交表格请点击圆形按钮”，这样的提示语对那些形状识别障碍人群来讲是没有帮助的，甚至会带来困惑。一个合格的测试人员要能够发现问题，也要能提供改进意见。那么针对这个问题，期望行为应是什么样的呢？首先：我们应给这个按钮加一个标签，如“OK”；其次引导语应修正为“提交表格请点击圆形按钮：OK”。这样的话，即使不能识别形状，也可以通过标签“OK”找到相应的按钮。

音频 / 视频

当软件或应用中包含音频或者视频时，要确保除过音频 / 视频，软件还提供了其他方式来向用户提供等同的信息。

例如，软件提供了一段包含对白和声音（自然的或是人工合成的）的录音。那惯常的“可达性”的实现方法是：提供一份文档（文字信息），包含对白等，来传达等同的信息。

对于视频的可达性实现也是类似的要求。

键盘可达性测试

键盘可达性是指通过纯键盘操作来使用软件，主要是针对软件的某些功能实现只支持鼠标操作来讲的。之所以要求键盘可达性，是因为很多实现可达性的辅助技术是依赖键盘交互的。键盘可达性测试需要覆盖以下几个方面：

TAB 遍历：即所有的可操作部分（文字，标题，链接，按钮，输入框等等，装饰性图片除外）都可以通过敲 Tab 键来遍历。

SHIFT+TAB 回溯，当你用 TAB 把整个软件遍历一遍以后，要用 SHIFT+TAB 来验证是否可以将整个软件回溯一遍。

TAB 顺序合理：是指使用 TAB 键遍历软件的时候，跳转顺序要与软件的使用流程相一致。例如，完成某个操作有 5 个步骤，而且步骤间有先后关系，那么 TAB 跳转的时候一定要按照先后顺序跳转。在我刚开始接触可达性测试的时候，当时的产品设计没有充分考虑可达性的需求，我曾经不止一次发现跳转顺序混乱的情况。

聚焦可视化：是指当用 TAB 键跳转到一个控件，链接，标题……的时候，需要有肉眼可见的视觉效果让用户知道当前 TAB 跳转到什么位置了。通常的实现方法包括给在按钮周围显示一个高亮的边框，链接变色，文字信息四周有虚线边框等。



布局设计一致：为了提高用户浏览软件的效率，要求重复出现的界面元素在不同页面显示的位置和顺序是一致的。比如，一个视力有缺陷的用户主要依赖“搜索”这个功能来查找网站的内容，网站设计实现的时候，最好“搜索”在每个页面出现的位置是一样的，这样就方便用户很容易地定位和找到“搜索”。其实，不光是视力缺陷的用户，普通用户也有这样的预期，布局一致的设计更符合人们的认知和使用习惯。

区块跳转：很多残障人士主要依赖键盘操作软件，如视力缺陷人群很难使用鼠标，像前文介绍的，他们通过敲 TAB 键来遍历软件，并通过读屏软件获取相应的信息。这样的话，如果一个软件或应用里含有大量的重复信息，如成百上千个链接，当使用 TAB 键遍历的时候，就会淹没浩如烟海的链接里。对这些用户来说，软件的可用性就很差，甚至几乎是不可用的。可达性软件有一个要求叫区块跳转，可以在一定程度上解决这个问题。

下面举个例子来说明区块跳转是什么及如何实现。例如一些大的门户网站，把信息作如下分类：新闻、财经、体育、房产、时尚、科技……，每一个分类都可以被称为一个“区块”。那么在设计实现的时候基本做到以下几点就实现了区块跳转：

- 1) 每个页面顶部放置一个可以直接跳回网站主页的链接。
- 2) 每个“区块”的顶部放置一个可以直接跳转到“区块”结束位置的链接；
- 3) 在页面顶部放置可以直接跳转到其他“区块”的链接。

实现了上述三个要求，基本上就可以做到无论在任何页面都可以直接在区块间跳转。比如用户感兴趣的是“理财”话题，那么按照键盘遍历的顺序，他首先到达的是新闻版块，由于该话题不是他关心的，该用户可以选择直接跳转到新闻版块结束的位置，继续 TAB，就可以到达他感兴趣的财经版块了。借由这样的实现，通过键盘访问网站的用户无须遍历他并不感兴趣的新闻版块，节省了大量的操作时间，网站的可用性也大大提升。

对读屏软件的支持

为了满足视力有缺陷的用户使用软件的需求，可达性的另一个基本要求是能够支持读屏软件，也即软件传达的信息以及如何操作都可以被“读”出来。

JAWS 是一款使用非常专业的读屏软件，由 Freedom Scientific 出品，最新的版本 JAWS 2018 发布于 2017 年 10 月，感兴趣的读者可以通过如下链接下载并尝试使用。



<http://www.freedomscientific.com/Downloads/JAWS>

正因为 JAWS 有相当广泛的用户群，所以很多可达性软件也都支持 JAWS，不光文字性的信息要被读出来。一些控件，如按键，表单，输入框，甚至图片等也要能被 JAWS 识别并读出来——这可以通过给控件添加标签等方法来实现。

可达性测试的工具

像任何其他测试一样，可达性测试也有一些自动 / 半自动化的测试工具，比如 IBM Rational Policy Tester(RPT)用来扫描 Web 网页的静态内容并判断是否符合可达性要求；Firefox 的 Dynamic Assessment Plugin (DAP)无论对开发和测试来讲都是一款很强大的工具，可以用来检测色差对比度，跳转顺序等。一些大公司也会开发一些仅供内部员工使用的可达性测试工具，不对外开放。

不过与其他测试（如功能、性能）稍有不同的是，可达性的自动化测试并不能取代（或部分取代）手工测试，而只能作为一种辅助测试手段。例如 Web 的自动化测试扫描出来的结果并不能作为最终测试结果直接提交，而是需要由专职测试人员对扫描出的问题进行复审，结合产品的具体实现来判定这些问题是否真的是可达性 bug。由此足见可达性测试对专业知识的要求之高。

上文介绍了软件可达性的基本要求和测试点。当然，可达性的外延很宽泛，而且随着科技的发展，可达性的标准也在不停地完善和丰富。另外针对不同的应用，比如单机版软件，Web 应用或者移动终端，可达性的要求和测试点也不尽相同。有志于探索可达性测试的达人，通过进一步的学习可以发现更多有趣的知识和技术。而且随着学习的深入，你会慢慢意识到，所谓可达性并不只是为了满足“残障”人士的需求，它的很多要求都更贴合人们的认知方式和使用习惯，可达性的实现会给所有使用软件的人带来便利！



高级信息系统项目管理师考试经验总结

◆ 作者：桃子

距离 17 年下半年软考的高项考试已过去将近 2 个月时间，回想起准备考试的日子仍历历在目，现如今吸引眼球的东西太多，需要我们花精力处理的纷杂事情也堆满了难能可贵的休息日，电视剧里跌宕起伏的剧情，奔波于陪孩子补课的路上等诸如此类，以至于每周拿出半天时间去阅读去学习都是极其奢侈的事情，不知大家是否也深有同感，所以如何能够既省时又省力的通过考试，是我们准备考试前都应该了解清楚的事情，俗话说“不打无准备之仗，方能立于不败之地”说的就是这个道理。

一、备考选择

我们都知道要想通过考试的决定性因素是分数，我们只要把握住考点以及每年常考和必考内容通过考试基本上没有什么问题，掌握这些考点的过程就是学习的过程，目前学习的方式基本上分为 2 种：培训机构、自学。

参加培训机构一方面可以节省时间，只需跟着培训班的进度学习就行了，另一方面的好处就是本次不过可以承诺你过为止（或者 2 年 3 年不等），坏处就是需要大家掏出腰包了，目前我了解到的费用在 1000-3000 不等，当然不要指着参加培训班自己私底下就不看书了，二者结合效果会更好，如果时间有限，自学能力差的考生，我建议选取一个性价比高的培训班是明智之举；那么对于选择自学的考生来说需要自己去理解的东西多一点，时间会稍微长一点，同时查找有用的资料也需要占用一定的时间。

二、选择自学如何收集资料学习

如果你认为自学才是证明自己能力的方式，从而增加自信心，笔者当时也是抱着这样的想法，明明有捷径偏要自己顶着头皮向前冲，那么有什么好的方法快速收集资料呢，下面我们就来说一说。



2.1、买书赠视频、学习资料

我首先在淘宝上购买参考书，比较了一下是否正版、价格差异等因素，其中有很多卖家都是提供视频学习资料的，可以客服小窗了解一下视频资料是否是近期最新的，购买即可

2.2、选择培训串讲视频

现在市面上网络的学习平台很多，有一些刚刚成立的培训机构需要招揽生源，定时推出免费的串讲视频，可以找来跟着学习，一方面了解自己学习进度，再者可以回顾自己知识点查缺补漏，可以添加 QQ 群，里面会有很多资料，比如历年真题、案例解析方法等，一般到考试前 2-3 个月就会有串讲视频了。

2.3、使用资料的先后顺序

现在我们有了参考用书、视频、培训班串讲视频，怎么利用比较好呢？我是首先结合大纲读书里的每一部分概要，知道了大致考试内容及重点及考试内容，其次根据重点内容听取了资料视频，一边听视频一边结合看书的详细内容（书很厚不建议每一章都看），这样一轮下来基本上对考试的 70% 内容都有所了解，最后临近考试跟着培训班听串讲视频，巩固知识点，相信到此阶段大家定会胸有成竹。此外，最好在学习过程中记笔记，有很多东西都需要记忆背诵的，常常翻出来看看，加深记忆。

每一科目有针对性备考

前面介绍了根据时间顺序如何纵向备考，大家都知道考试总共分为‘上午单项选择题’、‘下午案例分析题’、‘下午论文’。那么对于每一科目有哪些需要注意的事项呢，下面我们来介绍下横向学习。

3.1、上午单项选择题

对于单项选择来说，涉及知识点比较杂、乱，几乎每一项内容都会出考试总共 75 道题，答对 45 道为及格，难度为中等偏下，一般人都会通过，我没有放太多时间在这部分，可以利用信官网的每日一练功能，每天练习 10 道题，错误的添加到收藏夹内，临考 2 周前进行巩固就可以了

3.2、案例分析题

对于案例来说，2 大题型：计算和简答，每年都会必考进度及成本（关键路径和增



值管理), 简答题考察的是九大项目管理相关知识点, 常考什么原因引起的项目 xx 问题, 有什么解决方案, 你是项目经理应该如何做等, 我对于这部分就是做历年的真题, 总结规律, 比如回答的时候想想管理中是否缺少 x 机制, 缺少 x 策略, 缺少 x 流程, 案例中的做法不正确, 缺少 x 记录等。可以说如果计算题能答满分, 基本上下午的案例析题差不多就能过了。

3.3、论文

论文希望大家认真对待, 要想通过整个考试论文是关键, 需要在 2 小时内写 3000 字文章, 这个阶段一定要动手写, 我当时写了大概 4 遍, 建议每一篇都准备, 在考试前背写你认为最可能考试的文章。即使没有压中, 那些准备的摘要 (330 字)、背景 (600 字)、结尾 (400-500 字) 都是可以复用的, 关于内容部分按照九大管理框架往里填充, 凑凑字数还是够用的, 把握字数清晰, 字数够用, 文章结构清晰, 最后内容里添加自己的真情实感, 基本上几个没有问题

最后举一下我最容易犯的错误, 在我看完书和视频理解相关知识点后, 没有及时对重点内容进行背诵, 造成做题的时候只知道有这部门内容, 但不知道怎么选择或简答, 所以这可能也是其他人容易犯的错误, 对知识点消化吸收成为自己的才算真正意义上的掌握。上面就是我的学习心得, 希望对大家有所帮助。



知道这些，轻松处理临时任务

◆作者：桃子

即使有着丰富测试经验的测试人员，听到“临时任务”都会望而却步，更别说刚刚从事测试工作的从业者，那么我今天就带领大家梳理下临时任务特点，并结合自己的亲身经验总结出注意哪些方面能够快速、轻松的处理这些问题。

临时任务特点

1、时间紧任务重

最近一段时间公司要研发一款 APP，需要用到人脸识别技术，所以前期需要对市面上常用的几款有关人脸识别技术的应用进行调研，从而选取一款性价比较高的应用。因为涉及到市场、商务、研发各个部门，时间紧任务重，所以每接到一款新的可测试 app，都需要在 1-2 天内测试完成并提交报告。

2、没有《需求设计说明书》等相关文档

当我接到可测试的 APP 后，一般来说都没有《需求说明书》、《测试用例》等相关文档，也没有关于软件介绍的说明文件，我们只能结合需要做的产品与待测 APP 相关的功能总结测试功能点，比如说一体机进行人脸识别时需要考虑到距离是否符合要求、人脸比对时是否支持活体检测等。

3、没有 Bug 跟踪系统

一方面所测试的 APP 都是非本公司研发人员研发而成，无法搭建 BUG 跟踪项目；另一方面测试初衷是评估软件是否适配，所以没有必要重新建立项目跟踪记录，在这样情况下，我会将发现的问题总结到 excel 表中，写明序号、问题描述、问题步骤、复现概率、附件等相关内容，方便查阅，在发送测试报告时，统一发送给相关干系人。

我们需要具备哪些能力



对于类似这种临时的小任务，一般都是领导简单交代几句话，说明一下什么时候反馈结果，对比正常的流程我们会跳过需求理解阶段，直接进行测试，在测试过程中贯穿沟通、划定范围边界、总结报告等，那么下面我给大家介绍在哪些方面是需要我们平时注意积累的

1、突破心理障碍，提高快速应变能力

大家都知道，一个人在一个项目中呆久了，思维已经形成定势，面对熟练的功能和任务做起来可以得心应手；然而面对一个突发的临时任务，第一反应就是“我没有做过，我可以么？”这样在反问自己，我也曾经遇到过，有的同事第一反应脱口而出“我没有做过...”

2、加强沟通能力

沟通管理贯穿在项目始终，无论在与领导讨论任务时还是给测试人员分配任务时，可以说一个项目 70% 都在与人进行沟通。记得有一次我分配任务时，只告诉大家测试的功能点、范围有哪些，和每个人测试执行的用例有哪些，在测试过程中，发现有的人对测试功能的目的提出异议，后来我口头和对方沟通讲解了是由于接口改造会导致某些功能调用失败的情况，需要我们进行下全功能回归测试。通过这次事情，我想表达的是沟通不仅仅是表达自己的思想，也是了解其他人想法的过程，通过沟通解决疑问，更快速的高质量完成项目。所以说无论大小任务，平时多注意积累，总结自己沟通方面的欠缺，都是百利而无一害的。

3、总结报告能力

发送报告给干系人，可以使其了解当前任务状态，主要的成果物完成情况怎么样，当前严重问题有哪些，方便其作出决策。

在这里我想给大家提供 2 个建议：

1、因为时间特别紧张，我经常是手里有活的情况下还要完成另一个任务，都是没有时间进行测试用例编写的，但是发送报告时还是需要总结出测试场景，所以慢慢的就养成了边测试边记录测试场景的能力，这样做的好处是减少了测试问题总结的时间，你可以在测试结果处直接标记；

2、发送报告时，在详细问题前面用几句话简单介绍一下当前严重的问题有哪些及你认为当前版本质量，这样一来相关干系人可以避免从测试记录项中逐条发现当前问



题，为决策者节约时间，对项目质量有个心理预期。

对于这种临时的小任务，最主要特点是时间紧张，没有熟悉过需求，所以我们平时多注重自己应变能力，总结报告能力，和其他人沟通能力的锻炼，有心者事竟成，相信慢慢的你会发现一个不一样的自己。



愿有岁月可回首，且以勤奋共进步

◆ 作者：梁 凯

马上快过新年了，回首这一年，经历了很多的事情。

工作内容：

- 1) 对普华、华为、VMWare 三款产品的虚拟机性能进行测试和对比。
- 2) ServerOS 3.2 x86 测试，我主要负责自动化功能测试、ltp 功能测试、性能测试的部分。这是我正式将 isoft-ltp 测试工具用于实际工作中，因为之前一直处于学习的状态。
- 3) 龙芯 Server 5.0 测试，比 ServerOS 3.2 x86 开始稍微晚一点，工作内容上差不多，不再赘述。需要补充的是从这个项目开始增加了几个自动化测试用例。
- 4) 龙芯 Server 5.0 在 sugonL840-G10 的适配，主要是进行了硬件兼容性测试和性能测试。此时开始编写性能自动化测试脚本。
- 5) 龙芯 Server 5.0 名录测试，这段时间经常去五所出差，协助测试。
- 6) 海光服务器的适配测试，主要是硬件兼容性测试和性能测试。
- 7) 对 isoft-ltp 进行升级，增加测试用例脚本。

总结：

如果这一年，说一个工作关键词的话，那必是"出差"了。今年的市内出差频率太高了，刚查了一下月历，有记录的就 33 天，按工作时间算的话就是近 7 周。要么去曙光要么去五所。谁也不想去市内出差，这是一个事实，毕竟去一个陌生的公司，跟人家又不熟，来回路程、吃饭、喝水甚至有时去厕所也不方便的，只能说习惯就好了。现在我跟曙光的人员混的也很熟了，毕竟经常见面了嘛，有一次曙光的人员调侃说我是曙光的



半个同事了。其实出差还是有很多好处的，对其他公司的环境、企业文化、为人处世等方面都可以有所学习。最关键的是曙光有免费的午餐啊，哈哈！

对系统自动化测试经历了由了解到熟悉到掌握的阶段。对于自动化测试开始还是比较麻烦的，遇到问题你得看脚本，得调试，脚本不完善了还得修改脚本。但是一旦一两个小版本过去后就熟悉了，而且测试所用时间会越来越来少。测试效率会越来越高。

编写性能自动化测试工具--白菜自动化测试工具。其实 autotest 也可以完成性能自动化测试，但是它仅限于在公司内部使用，出差就没法用了。于是我自己尝试写了一套性能自动化测试工具，主要是用 shell 和 python 写的。它的特点是携带方便、单机运行、免安装。

性能测试工具的安装、运行、收集测试结果、发送测试结果到邮箱全是自动完成。

分享 python 编程培训。把自己学习过程的经验分享给大家，一方面看看自己到底学到了多少东西，一方面看看自己学到的东西能不能讲给大家。希望与大家共勉，共同进步。发现很多同事都在学习、使用 python，有的同事还写了很多好玩的脚本。能够有一种学习的氛围其实蛮好的。一直希望大家多学习专业知识，提高工作效率。

读的书有《Linux 命令行与 shell 脚本编程大全》、《C++程序设计》、《C 程序设计》、《python 学习手册》。应该说读的不算多，至少比去年少好几本呢。一方面大概是因为这一年出差比较多，就没时间读书了；另一方面是 c、c++这两本书几乎用了大半年的时光，光 c 语言估计得近半年，我啃的非常的艰难，进步也很慢。书中的大部分例题我都自己敲了一遍。也算机缘巧合，在学习的同时正好有个朋友找我，他在国外做博士，选修了 c 语言课程，其中的作业有不会的就让我做，这给了我一个练手的机会，像字符串处理、编写 shell 命令等，对 linux 底层的编程有了一些了解。但我感觉自己现在的水平充其量算是入门。C 语言很底层，很基础。像内存操作、指针、数组、各个系统函数的调用等等，真的是需要花功夫去慢慢的学习的。总之，c 语言想说爱你不容易。

感悟：

同学，你今天堕落了吗？

这是我想问每一个同事的一个问题。我发现我们公司某些员工有堕落的趋势，而且员工业务水平也呈现下降的趋势。尤其是新员工，本来入职时水平就不是特别好。开始时到是比较勤奋好学，可是过了一段时间后，就堕落了，我评判的标准是：上下班的时



间、空余时间自学的程度、对任务的重视程度、对任务的投入量。说别人也是在说我自己，我感觉我自己也在堕落，可能稍微轻一些，大家不易察觉而已。有时我在扪心自问：难道我要浑浑噩噩的去工作吗？每天9点多上班、5点半一过就下班、困了就睡一觉、领导给的任务应付一下就行了。我想这样做不仅害了公司还害了自己呀！我们不能做三等（等下班、等放假、等工资）公民呀！我们还年轻，我们有的是时间和精力，为什么不用来提高自己的业务水平呢？以备将来在竞争的环境里立于不败之地。

我在 Python 编程的分享中就提到过，还是希望大家每天拿出一点时间来学习，提高工作能力。须知投资学习是一件只赚不赔的生意。若有恒，何必三更眠五更起，最无益，莫过一日曝十日寒。希望大家不忘初心，砥砺前行。

兴趣是最好的老师

当初在南开大学面试时，晓宇姐曾问我一个问题：“你对计算机行业工作的印象是什么？”我说：“经常加班？很枯燥？”晓宇姐说：“如果感兴趣的话就不会感觉枯燥了。”想起当时的场景，至今历历在目。是呀，兴趣是最好的老师，人一天三分之一以上的时间在工作，可以想象一下，如果每天很讨厌工作、讨厌上班的话是一件多么痛苦的事情呀。而我认为兴趣跟成就感是密不可分的，当一件工作苦思冥想不得其解，经过一番努力，柳暗花明的那一刻是很兴奋的也很有成就感的。正是这份成就感在支持着一个人快乐的工作。所以我们应该时时给自己一个挑战，让自己得到一份成就感。

大学之大在大师，公司之强在强人。

一座学校的好坏并不在于校园环境有多优美，硬件设备有多强大，而在于师资的力量。同样一个公司强不强大主要还是看有没有人才，每个公司都不缺人，但缺人才。有人才就会带动公司进入良性循环。我认为“三个臭皮匠，顶一个诸葛亮”用在公司用人上是不成立的。宁可辞掉十个臭皮匠也要留住一个诸葛亮。所以一个好的公司会花大力气不惜代价的去寻找人才，更关键的还要留住人才。

经验的传承很重要。

对于一个问题，各种网上查资料，各种做尝试，我两天都没有解决。这时一个同事过来了，简单听了我的描述后，一句话就解决了。这是什么？这就是经验的力量！可能他专业水平并不一定多高，但是他做过，他遇到过这类问题，就是说他有经验。在学习知识这方面我认为看书不如看视频，看视频不如听老师当面讲课，听老师讲课不如一个



老师手把手的带你。

我们组在这方面就做出了努力，某某同事在做完一个项目或者一个调查后会要求写个总结。这就是一种让经验得以有效传承的方法。其实我建议一个员工在入职时能安排一个老员工来手把手的带。老员工对新员工负有责任，要定期的沟通。我刚入职那会遇到问题就不好意思去问别人，因为别人没有义务帮助我。一旦给我安排一个师傅就解决这个问题了。

未来计划

1) 升级完善自动化测试、增加自动化测试的覆盖率。

将 autotest 服务器的系统升级到 ServerOS 4.0 上，并增加其他的网络服务，协助测试。完成就的测试用例，增加新的测试用例。

2) 学习专业知识、编程知识

对专业知识要深入学习。编程是基本功，也要深入学习。语言方面还是以 Python、c 为主。对界面、多线程、网络套字节等等要深入研究。

结语

我们终将老去，愿老有所得、老有所值，愿有岁月可回首。

《51 测试天地》(四十八) 下篇 精彩预览

- 测试女巫之做个有灵魂的测试师
- 定时重进程的 Shell 脚本实例
- 从测试开发到 Delivery Manager
- 软件测试行业是一座围城
- Eclipse+appium+夜神虚拟机环境调试
- 职场小白的职业探寻之路

马上阅读

