360技术嘉年华

搜索接口功能自动化测试工程实践

功能自动化、持续集成&线上监控

董十月

360搜索技术经理

服务端测试开发专家

360

目录/CONTENTS

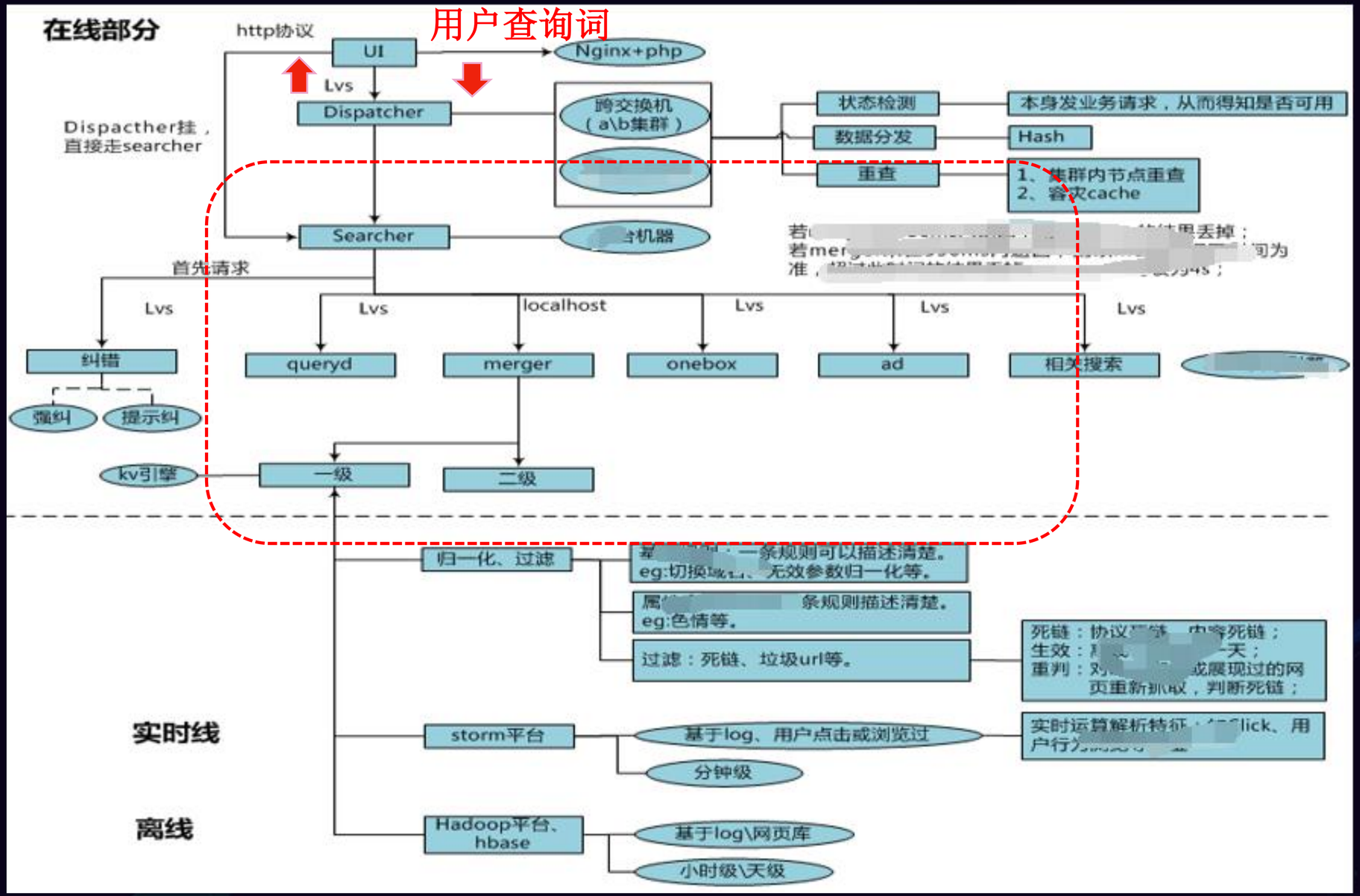
Part 01

接口功能自动化的背景和框架设计思路

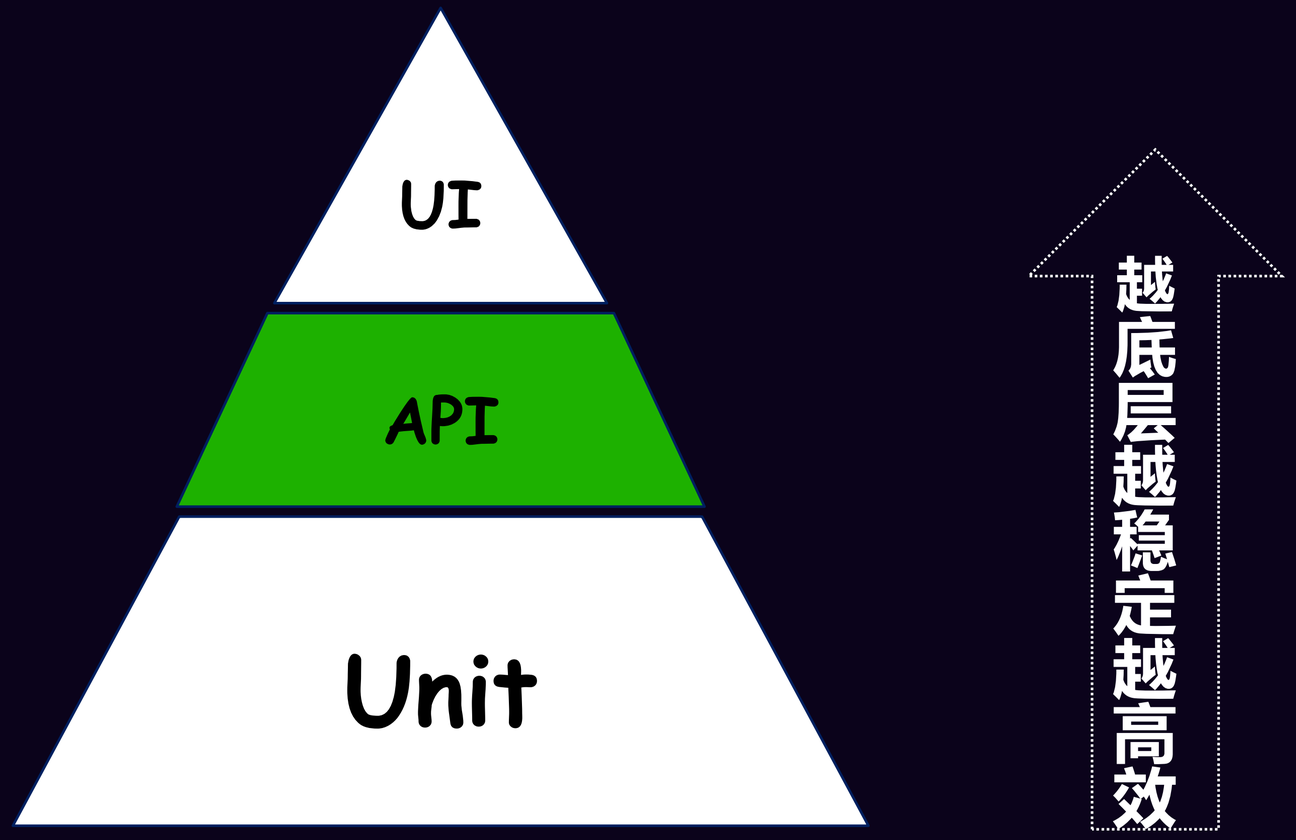
搜索服务架构简介

- 什么是接口?

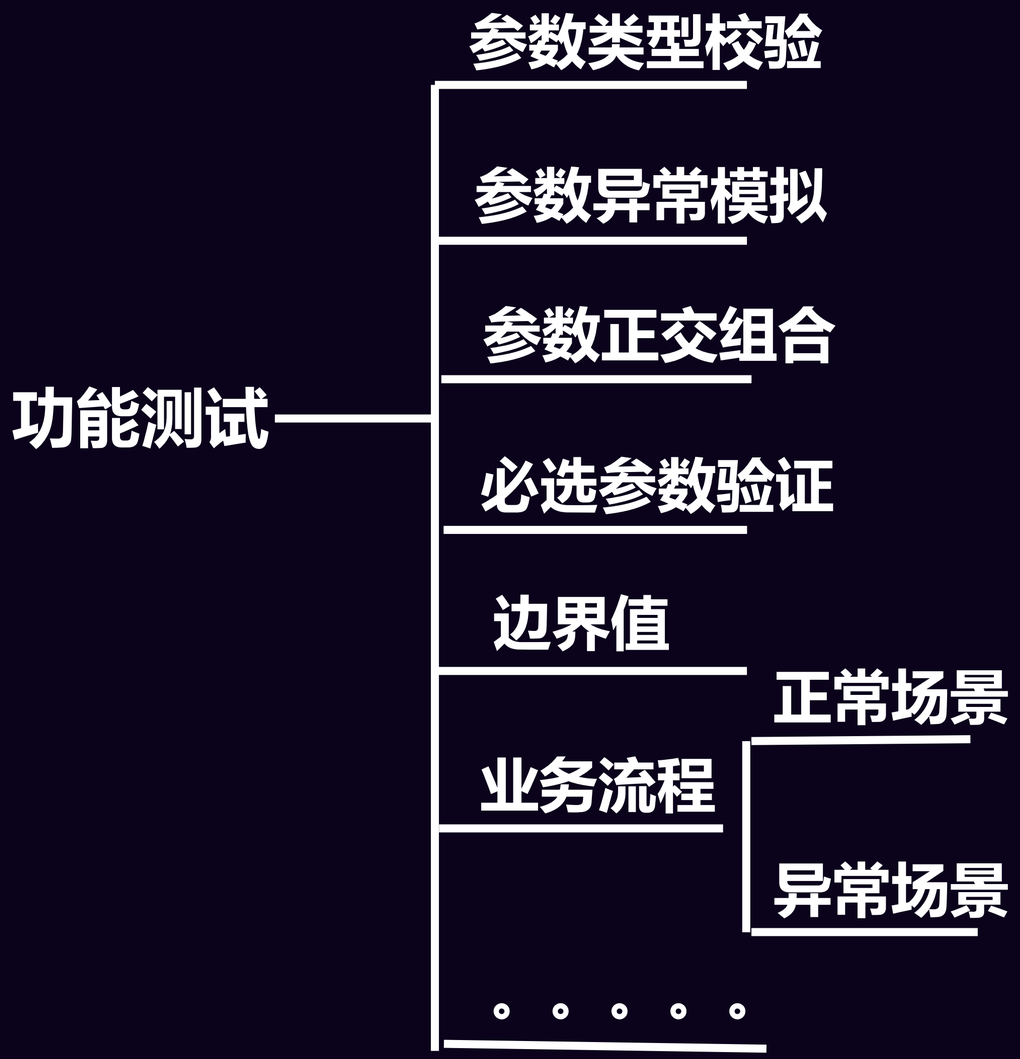
服务模块间沟通



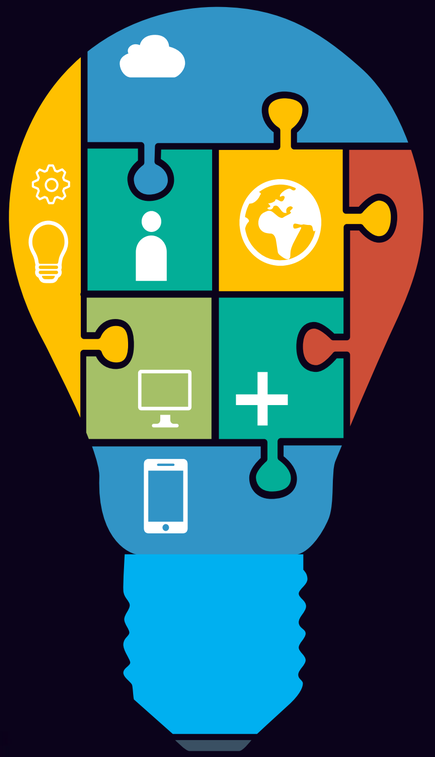
为什么做接口测试



接口功能测试的策略



接口功能测试方法&工具



01 **在线调试工具**

浏览器

02 **常用工具**

Fiddler、Postman、Soupui、Jmeter

03 **编写代码**

Python+urllib2/requests、Java+Httpclient

为什么选择写代码实现接口功能自动化?

- 加密、带签名(sign)接口
- 多接口组合调用
- 可扩展性(线上监控、定制邮件)

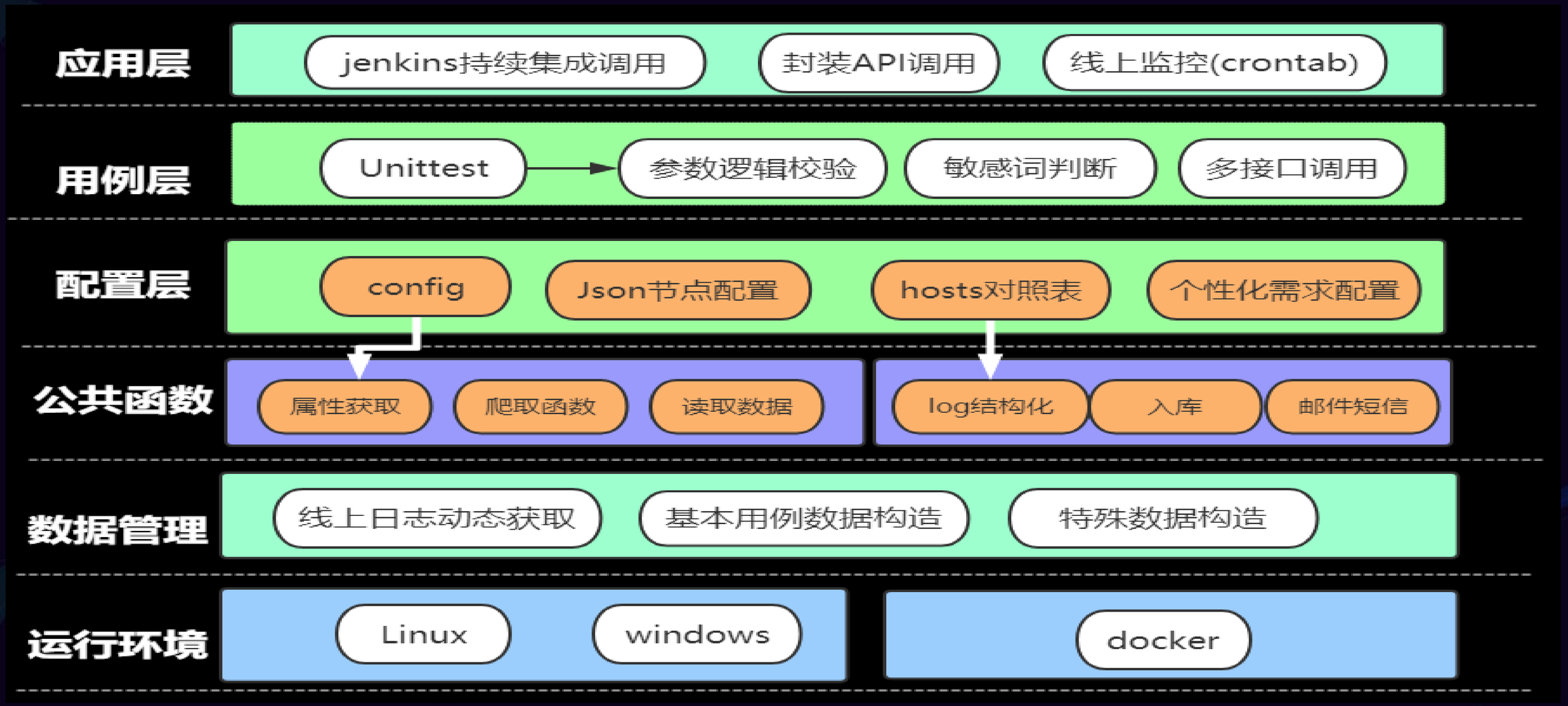
Auto_ApiTest

Auto_ApiTest自动化工具核心特性

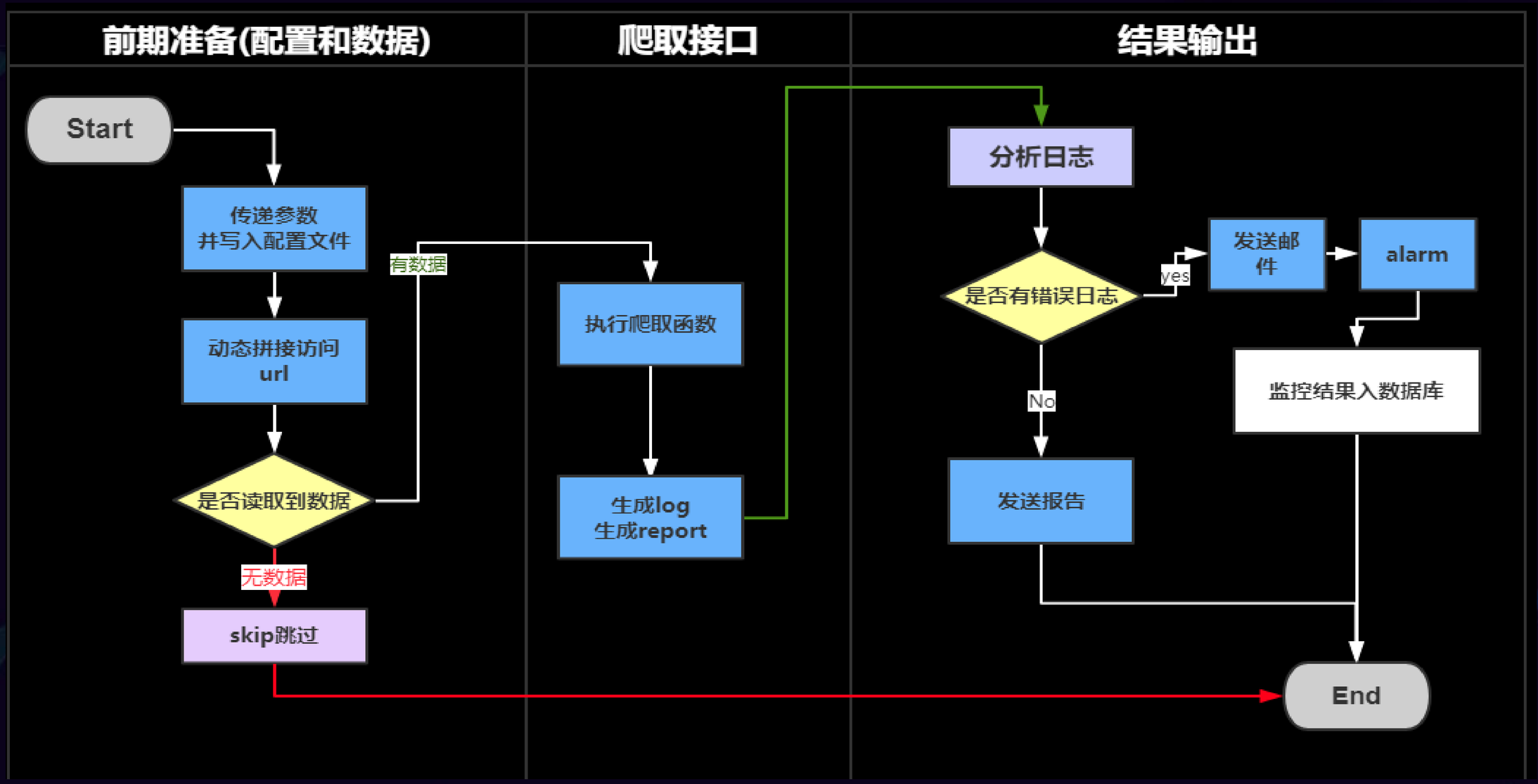
1. 支持API接口的多种请求方法
2. 数据与代码分离
3. 支持线上监控报警
4. 测试执行方式简单灵活
5. 测试报告简洁清晰

Auto_ApiTest工具分层图

◆python+unittest+requests+shell



Auto_ApiTest工具执行流程

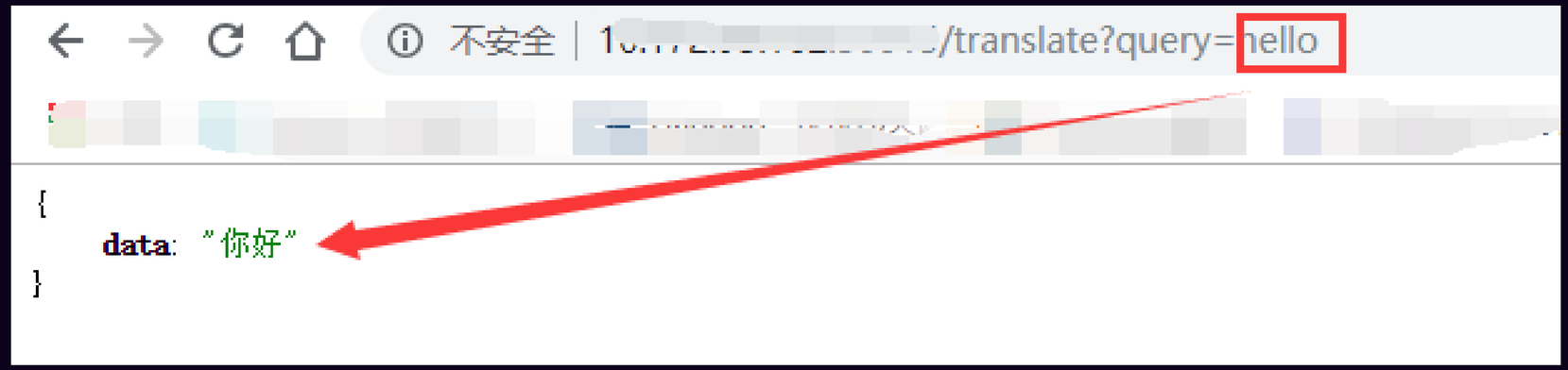


Part 02

核心功能实现方法详解

Auto_ApiTest数据管理--数据定义

- 哪些数据才需要管理？如何存储？




```
query,expect
hello,你好
apple,苹果
how are you,你好吗
nothing is diffcult,没有什么是困难的
It cost me a $12 taxi ride,它花了我12元的出租车费
```

Auto_ApiTest数据管理--数据特征

- **普通数据**

  有固定的输入输出

- **特殊数据**

  需要动态获取后格式化成可用的数据格式

数据管理层-搜索接口数据构造方法

◆ 普通数据-手动构造

```
1 query,city,mp,exceptname1,exceptname1id,exceptname2,exceptname2id
2 10号线,北京,116.501004:39.990192,地铁10号线外环,b9e6e78047b9a513,地铁10号线内环,
  3c2cab34cecd8041
3 1号线,天津,117.209764:39.093709,地铁1号线,5fba1c733fdf6c81,地铁1号线,39b80ee2ea1328ab
4 1号线,西安,108.946107:34.347281,地铁1号线,683ba70ac30f63a0,地铁1号线,05386e732520f98b
5 1号线,广州,113.271283:23.135288,地铁1号线,b162afb42bc116ae,地铁1号线,33dc2de1a295f6d2
6 4号线,上海,121.480524:31.23595,地铁4号线外环,b164050724a9c9c4,地铁4号线内环,
  7e81e55aebb1b9ee
7 地铁1号线,深圳,114.06444:22.548488,地铁1号线(罗宝线),79dc625835d3c866,
  地铁1号线(罗宝线),76c6dde37bd29fc4
8 402路,北京,116.501004:39.990192,402路,d54fc1e1bcc2a8f2,402路,b52473efc4b528d9
9 851路,南京,118.802759:32.064702,851路,5656aa77626719e1,851路,eb7548b7ac7a3c15
10 102路,郑州,113.631732:34.753459,102路,7279534e0f4314b3,102路,9ae684007ba916fa
11 202路,哈尔滨,126.541527:45.808873,202路,3d1c15d36b134ace,202路,f20f619d81129e84
12 109路,沈阳,123.437191:41.810777,109路,31dbee18401f42ef,109路,2d332a43946c7564
~
```

数据管理层-搜索接口数据构造方法

◆ 普通数据-线上日志获取

```
${HADOOP_CMD} streaming \
-D mapred.reduce.tasks=5 \
-D mapred.job.name="${USER}_${PREFIX}" \
-D mapred.job.priority=VERY_HIGH \
-D mapred.job.max.map.running=50 \
-D mapred.job.max.reduce.running=50 \
-input ${INPUT} \
-cacheArchive hdfs:///home/mr/lib/python2.7.tgz#python \
-output ${OUTPUT} \
-mapper "python/bin/python mapper.py" \
-reducer "python/bin/python reducer.py" \
-file mapper.py \
-file reducer.py
if [ $? != 0 ];then
#python alarm.py
exit 0
fi
```

数据管理层-取线上日志构造测试数据(根据会后疑问的补充信息)

举个例子：下面图中的数据为线上实时获取数据，在生成数据脚本中已经加入了判断标准如mip:或者amp:等，所以如果想用线上获取的数据作为测试用例数据，必须有预期值并且这个预期值是可以自动生成的，否则线上获取的数据会存在如你们所问对于结果验证是否合理的问题

```
    if mark == "mip":
        print "mip:%s"%amp_url
    elif mark == "amp":
        print amp_url
```

```
1  query
2  "mip:http://mip.findlaw.cn/ask/question_jx_3690.html"
3  "mip:https://mip.findlaw.cn/ask/question_19896628.html"
4  "mip:https://mip.findlaw.cn/ask/question_22274668.html"
5  "mip:https://mip.findlaw.cn/ask/question_22584104.html"
6  "mip:https://mip.findlaw.cn/ask/question_23475874.html"
7  "mip:https://mip.findlaw.cn/ask/question_25076564.html"
8  "mip:https://mip.findlaw.cn/ask/question_25298828.html"
9  "mip:https://mip.findlaw.cn/ask/question_27137840.html"
10 "mip:https://mip.findlaw.cn/ask/question_27729702.html"
11 "mip:https://mip.findlaw.cn/ask/question_28487856.html"
```

数据管理层-搜索接口数据构造方法(根据会后疑问的补充信息)

接上面的例子，我们来看下用例编写时候如何验证这些数据：

```
        content = content.decode(charset).encode("utf-8")
        htm_type = ""
        try:
            second_line = content.split("<head>")[0]
        except Exception, e:
            message = "mip标签按照head分割失败"
        else:
            htm_type = "mip" if "<html mip" in second_line or "mip>" in second_line or "mip=" in second_line else "amp"
        if except_type == htm_type:
            test_result = "Pass"
            break
        else:
            content = second_line
            message = "格式不正确query显示为%s但content中目前为%s"%(except_type,htm_type)
    elif i != 2:
        continue
    else:
        message = "content内容为空errorno=%s|statuscode=%s|response_time=%sms"%(errorno,status,response_time)
elif i != 2:
    continue
else:
    message = "接口访问失败未返回数据statuscode=%s|response_time=%sms"%(status,response_time)
```

mip标签作为判断使用

数据管理层-搜索接口数据构造方法

◆ 特殊数据之时效性数据-业务接口动态获取

```
#视频数据动态更新
function getshortquerys() {
    local filepath="../data/video_short_video"
    local filepath_short_so="../data/video_short_so"
    querys=`curl http://x.x.x.x/video/smallvideo/small_video_query.txt |shuf -n 100
    >$filepath/test_resultbak.txt`
    if [ $? -eq 0 ];then
        sed "1i query"
    $filepath"/test_resultbak.txt" >$filepath"/test_result.txt"
        if [[ $? -eq 0 ]];then
            cp $filepath"/test_result.txt" $filepath_short_so"/"
        else
            echo "cp fail"
        fi
    else
        echo "wordseries update error"
    fi }
```

数据管理层-通过接口获取数据落地(根据会后疑问补充信息)

如获取视频数据，自动获取数据格式如下，注意只有query一列

	query
1	大爱无敌
2	白气球
3	一地花香
4	时尚秀感
5	哈文：春晚背后的故事
6	十二金牌
7	仇恨的代价
8	疯丫头第一部
9	黄小鸿之狮舞骄阳
0	老爹的非“城”勿扰
1	幸福总动员
2	铸就雪域军魂六十年
3	乡路
4	科普月·奇妙世界我知道
5	培乐多超大奇趣蛋惊喜蛋
6	眼斗

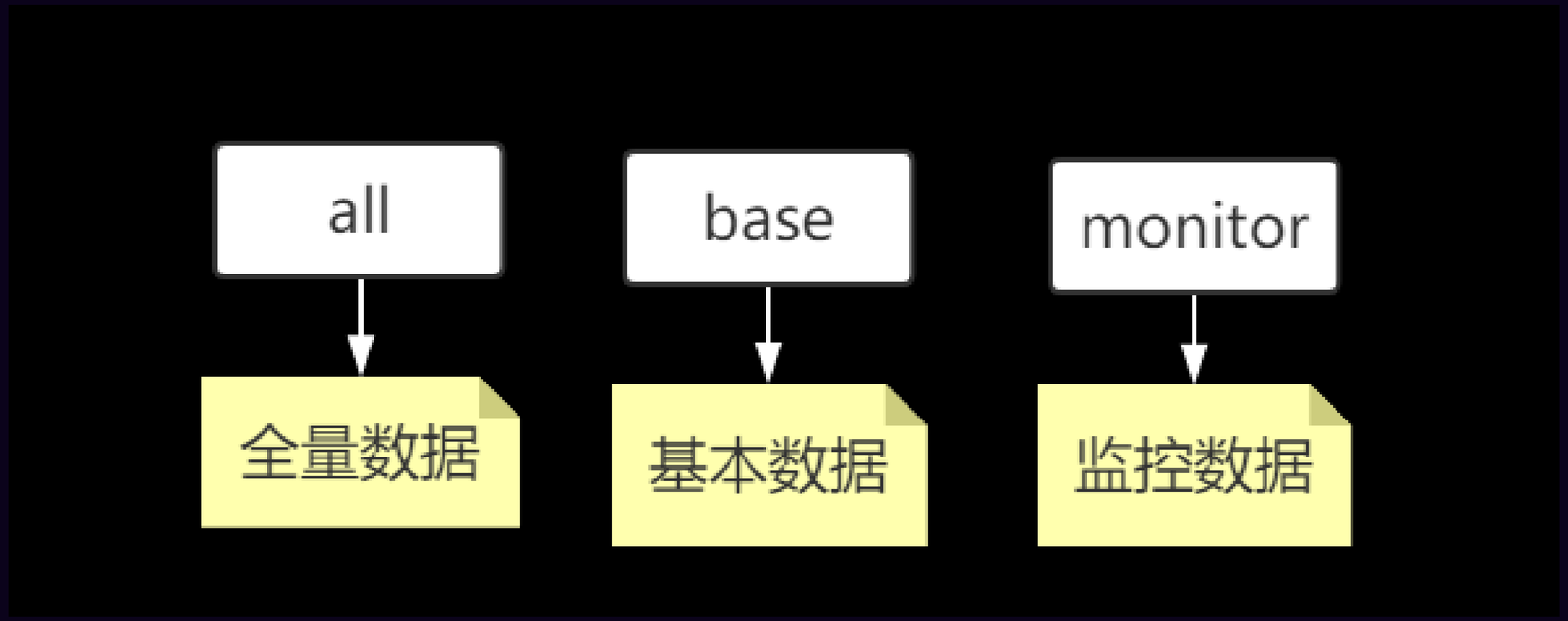
数据管理层-通过接口获取数据落地(根据会后疑问补充信息)

接上面例子：生成数据为何只有query一列，预期值怎么知道的，对于视频数据只是要验证接口返回结果的某一个节点是否是list类型，并且长度大于1，在逻辑判断时判断即可，当然也可以自动生成

```
allurl = return_dataall.get("url")
status = return_dataall.get("status")
response_time = return_dataall.get("response_time")
long_result = commonMethod.dict_get(return_data,data.DataResult.videos)
if return_data:
    if long_result:
        if len(long_result) > 1 and isinstance(long_result,list):
            test_result = "Pass"
            break
        else:
            message = "%sresult结果类型为%s有%s个元素|结果正常应该大于且类型为List"%(apiname,type(long_result),len(long_result))
            break
    elif i != 2:
        continue
    else:
        message = "%sresult结果返回为空responsetime=%s"%(apiname,response_time)
elif i != 2:
    continue
else:
    message = "%s访问失败未返回数据status=%s|response_time=%s"%(apiname,status,response_time)
```

数据管理层-数据目录设计

- 监控数据和自动化数据分离
  - 以接口为单位分别存放



配置层

- Config.ini&ConfigParser

1）通用配置，如：线程数、log存储地址、端口号等

2）单个test_*测试函数特性配置，如：通过百分比等

```
[DEFAULT]
comment = 5
service_inner = keyword_searcher
log_root_dir = log/%(service_inner)s
newest_log_dir_name = 1563244222
newest_log_dir = %(log_root_dir)s/%(newest_log_dir_name)s
hosts = [redacted]
data_root_dir = data
data_dir = %(data_root_dir)s/%(service_inner)s
current_domain = [redacted]
hostport = [redacted]
server_url = [redacted]
error_log_file_path = %(newest_log_dir)s/%(service_inner)s_%(newest_log_dir_name)s_error.csv
summery_log = %(newest_log_dir)s/summery.csv

[test_result]
comment = 检查result结果数量
section_name = test_result
```

配置层

●Data_config.py

```
#取json中需要的数据
class DataResult:

    #图搜searcher接口
    start = "/start"
    poi = "/poi"
    count = "/count"
    content = "/content"
    busline = "/busline"
    startkey = "/startkey"
    endkey = "/endkey"
    startpoi = "/startpoi/0/name"
    endpoi = "/endpoi/0/name"
    startcount = "/startcount"
    endcount = "/endcount"
    fold = "/fold"
    startcount = "/startcount"
    endcount = "/endcount"
    #图搜节点
    result = "/result"
    safe = result + "/0/safe"
    source = result + "/0/source"
```

json节点配置

```
class TitleName:
    report_titlename = {
        "Video_long":"长视频接口",
        "Nlp_history":"历史词接口",
        "Base_map":"底图接口监控",
        "Wenda_merger":"问答merger接口监控",
        "Zhanzhang_platform":"站长平台接口监控",
        "Onebox_dataapi":"onebox dataapi接口监控",
        "Video_onebox_merger":"视频onebox_merger接口监控",
        "Video_short_so":"搜索短视频接口监控",
        "Video_short_video":"影视站短视频接口监控",
        "Video_youtube":"视频youtube引擎监控",
        "Nlp_correct":"纠错服务接口监控",
        "Music_engine":"音乐引擎接口监控",
        "Nlp_recall_server":"Nlp新闻推荐url召回引擎监控",
        "Nlp_360recom_mobile":"为您推荐Mobile接口监控",
        "Nlp_embedding_search":"推荐向量服务接口监控",
        "Nlp_urlrec":"url点击推荐mobile&pc接口监控",
        "Nlp_related_search":"相关搜索接口监控",
        "Nlp_browser_recom":"推荐浏览器推荐服务接口监控",
```

测试报告标题对照表

```
}
host_jifang = {
    "[illegible]":"shbt",
    "[illegible]":"zzzc",
    "180.163.251.243":"shbt",
    "[illegible]":"zzzc",
    "[illegible]":"shm",
    "[illegible]":"shyc2",
    "[illegible]":"zzdt",
    "[illegible]":"zzzc",
    "[illegible]":"shbt",
    "[illegible]":"shbt",
    "[illegible]":"shyc2",
    "[illegible]":"zzdt",
    "[illegible]":"shbt",
    "[illegible]":"zzdt",
    "[illegible]":"zzzc",
    "[illegible]":"shyc",
    "[illegible]":"shbt",
    "[illegible]":"zzdt",
    "monebox03.se.zzdt.gihoo.net
```

机房IP对照表

配置层-补充会后疑问

- 节点获取使用dpath模块，具体方法如下：

```
#获取json内容节点
def dict_get(obj,glob,separator='/'):
    value = None
    try:
        value = dpath.util.get(obj,glob,separator)
    except Exception, e:
        pass
    return value
```

配置层-补充会后疑问

- 节点获取方法调用如下：

```
status = return_dataall.get("status")
response_time = return_dataall.get("response_time")
content = commonMethod.dict_get(return_data,data.DataResult.content)
errorno = commonMethod.dict_get(return_data,data.DataResult.todaynewserrorno)
if return_data:
    if content:
        content = base64.b64decode(content)
        content = zlib.decompress(content)
        charsets = ["utf-8", "gbk"]
        charset = ""
        for c in charsets:
            try:
```

文件名-类名-属性(节点)

公共函数层-数据读取

- CSV

字典类型(DictReader)

List形式存储

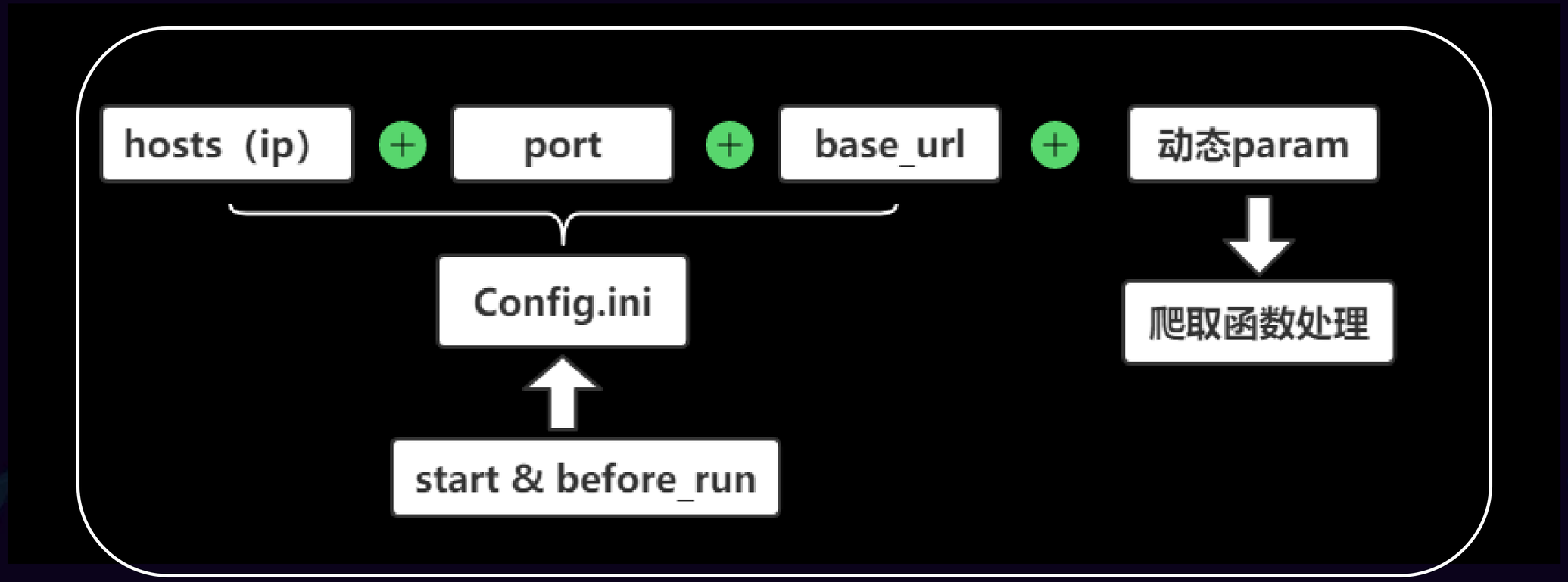
支持过滤带#号的数据

公共函数层-数据读取


```
 7 class DataReader:
 8     def __init__(self, data_file_path):
 9         self.data_file_path = data_file_path
10         self.data_file = open(self.data_file_path)
11         # 过滤以#开头的行
12         self.data_reader = csv.DictReader(filter(lambda row: row[0]!='#', self.data_file))
13         self.datas = []
14         for row in self.data_reader:
15             self.datas.append(row)
16 
17     def getRowCount(self):
18         return len(self.datas)
19         #return sum(1 for r in self.data_reader)
20 
21     def getFieldNames(self):
22         return self.data_reader.fieldnames
23 
24     def readCell(self, col_name, row_num):
25         return self.datas[row_num][col_name]
26 
27     def readRow(self, row_num):
28         return self.datas[row_num]
```

公共函数层-爬取函数

- **接口多样化---如何满足支持爬取不同类型的接口?**
  - url拼接方式



公共函数层-爬取函数

- 接口多样化---如何满足支持爬取不同类型的接口?
- 加密接口、带hosts访问

```
def getPostDateforNlp(base_url,data):
    lens = len(data)
    base64data = base64.b64decode(data)
    req_header = {'Host':'tip.f.360.cn'}
    url = base_url
    return_dict={}
    retryTime = 1
    return_dict["url"]=url
    doc = ""
    status = ""
    return_dict["data"]=""
    try:
        req = urllib2.Request(url, data=base64data, headers=req_header)
        res = urllib2.urlopen(req, timeout=3)
        doc = res.read()
        status = res.getcode()
    except Exception,e:
        print e
        return return_dict
    else:
        return_dict["data"]=doc
        return_dict["status"]=status
    return return_dict
```

解密

带Hosts方式

公共函数层-爬取函数

- 接口多样化---如何满足支持爬取不同类型的接口?
  - http/https/get/post

```
def getDataForGuoxue(base_url,query,req_timeout=0.15,retryTime=1):
    url = base_url + query
    return_dict = {}
    while retryTime > 0:
        try:
            req = requests.get(url,timeout=req_timeout,verify=False)
            response_time =  req.elapsed.microseconds
            status = req.status_code
            #return_dict = {}
            return_dict["url"]=url
            return_dict["data"]=""
            return_dict["status"]=str(status)
            return_dict["response_time"]=str(response_time/1000)
            doc = req.content
            return_dict["data"]=doc.strip()
            if(not return_dict["data"]):
                raise Exception,"reponse data is null"
            return return_dict
        except Exception, e:
            print e
            retryTime -= 1
            if retryTime == 1:
                print 'reTryTime:',retryTime,'\n\n'
    return return_dict
```

解决https访问鉴权问题

公共函数层-爬取函数

- 接口多样化---如何满足支持爬取不同类型的接口?
  - 通用接口

```
def getDataforNlp(base_url,**args):
    req_header = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1) AppleWebK
                  Chrome/39.0.2171.71 Safari/537.36', 'Content-Type': 'applic
    req_timeout = 1
    urlargs = ""
    url = ""
    for key in args:
        urlargs += "&" + key + "=" + args[key]
    if base_url.endswith("&") or base_url.endswith("?"):
        url = base_url + urlargs.lstrip("&")
    else:
        url = base_url + urlargs
    #print args
    s = requests.session()
    s.keep_alive = False
    return_dict = {}
    return_dict["url"]=url
    return_dict["data"]=""
    return_dict["status"]=""
    return_dict["response_time"]=""
    req=""
    try:
        req = requests.get(url,timeout=1,headers=req_header)
        response_time =  req.elapsed.microseconds
        status = req.status_code
        return_dict["status"]=status
        return_dict["response_time"] = "%sms"%(response_time/1000)
    except Exception,e:
        print e
    else:
```

不定长参数传递

测试用例管理

- **unittest**

  setup/setUpClass：获取执行的基本属性值如domain/port、日志格式等

  teardown/tearDownClass：运行结束后成功失败百分占比计算

- **case编写**

  以接口为单位

  逻辑代码处理、断言、和日志生成

测试用例管理-会后补充信息

- unittest使用，原报告生成功能保留，主要使用二次分析日志生成的报告

```
47 def creatsuite():
48
49     testunit=unittest.TestSuite()
50     #discover方法定义
51     discover = unittest.defaultTestLoader.discover(result_path,
52                                                    pattern='test_*.py',
53                                                    top_level_dir=None)
54     for test_suit in discover:
55         for test_case in test_suit:
56             testunit.addTests(test_case)
57     return testunit
58
59
60 if __name__ == '__main__':
61     testunit = creatsuite()
62    # file_path = '../../report/'
63     file_path = '../report/{0}/'.format(service_inner)
64     if not os.path.exists(file_path):
65         os.makedirs(file_path)
66     test_time = time.strftime("%Y%m%d_%H_%M",time.localtime(time.time()))
67     file_name = file_path + service_inner + domain.replace(".","_") +'_'+test_time + ".html"
68     fp = file(file_name,'wb')
69     runner = HTMLTestRunner.HTMLTestRunner(
70         stream=fp,
71         title=report_titlename[service_inner],
72         description=u'用例执行情况' )
73     test_result = runner.run(testunit)
74     sys.exit(not test_result.wasSuccessful())
75
```

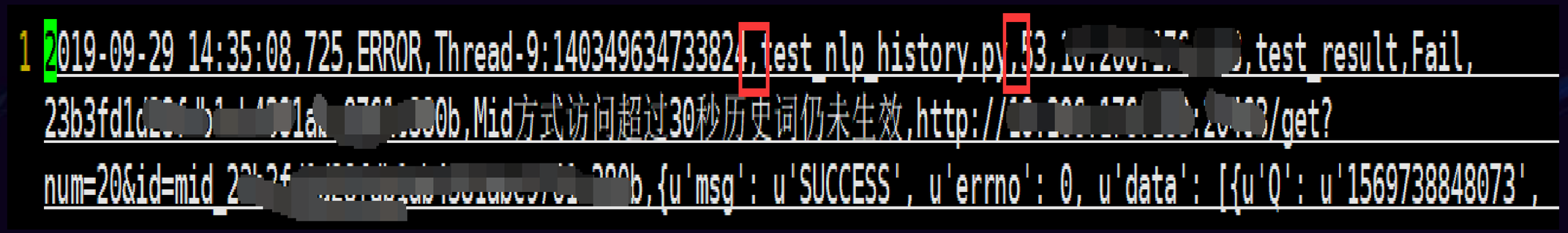
日志处理

◆ 日志结构化输出

● 代码格式

```
self.comm_log_str = '{0},{1}'.format(self.__class__.current_domain, self.section_name)
log = [test_result, query, message, allurl,return_data]
log.insert(0, self.comm_log_str)
self.logger.error(u'{0[0]},{0[1]},{0[2]},{0[3]},{0[4]},{0[5]}'.format(log))
```

● 结构化输出效果



日志处理-会后补充信息


```
                    test_result = "Pass"
                    break
                else:
                    content = second_line
                    message = "格式不正确query显示为%s但content中目前为%s"%(except_type,htm_type)
            elif i != 2:
                continue
            else:
                message = "content内容为空errorno=%s|statuscode=%s|response_time=%sms"%(errorno,status,response_time)
        elif i != 2:
            continue
        else:
            message = "接口访问失败未返回数据statuscode=%s|response_time=%sms"%(status,response_time)
    log = [test_result, query, message, pageurl,content]
    if test_result == 'Pass':
        self.pass_num += 1
        log.insert(0, self.comm_log_str)
        self.logger.info(u'{0[0]},{0[1]},{0[2]},{0[3]}'.format(log))
    elif test_result == 'Fail':
        self.fail_num += 1
        log.insert(0, self.comm_log_str)
        self.logger.error(u'{0[0]},{0[1]},{0[2]},{0[3]},{0[4]},{0[5]}'.format(log))
commonMethod.pool_map(fuc,self.datas, self.thread_num)
```

INFO和error日志主要用于二次日志分析生成邮件报告使用

日志处理-会后补充信息-断言问题

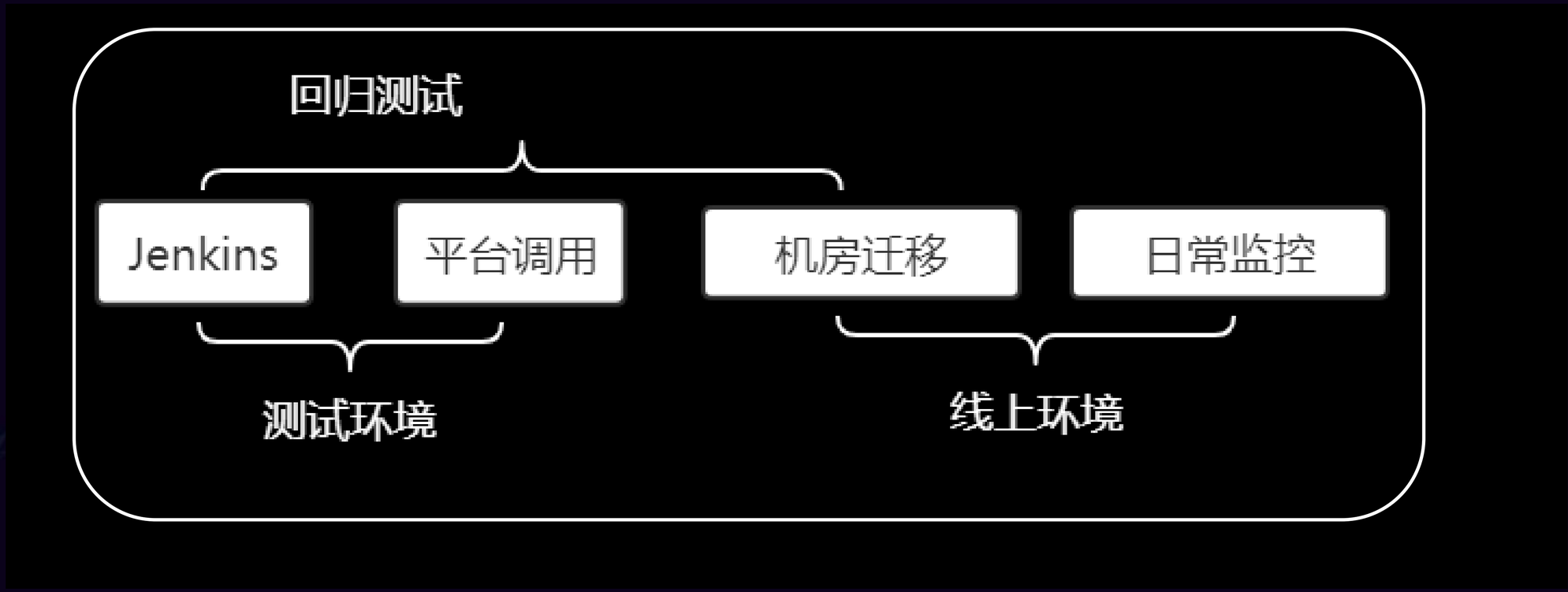
◆ 有些同学注意到用例中并未涉及到断言，是因为单个函数支持按百分比报错，所以在teardown中报一次即可，与二次日志分析相互独立的

```
def tearDown(self):
    self.total_num = self.fail_num + self.pass_num
    if self.total_num == 0:
        return
    self.fail_rate = float( '{:0.4f}'.format( (self.fail_num/self.total_num)*100 ) )
    self.pass_rate = float( '{:0.4f}'.format( (self.pass_num/self.total_num)*100 ) )
    print "f4444",self.fail_num,self.pass_num,self.fail_rate,self.expected_pass_rate
    self.test_result = 'Fail'
    if self.pass_rate >= self.expected_pass_rate:
        self.test_result = 'Pass'
        print "ok"
    self.assertGreaterEqual(self.pass_rate, self.expected_pass_rate)
```

执行层

- shell

传递各种参数，如端口号、Hostname、收件人、执行目录等



自动生成测试报告

- HTMLTestRunner

360地图接口自动化报告

Start Time: 2019-09-06 16:46:40

Duration: 0:00:39.668964

Status: Pass 13

用例执行情况

Show Summary Failed All

Test Group/Test case	Count	Pass	Fail	Error	View
test_keyword_searcher.Keyword_searcher	13	13	0	0	Detail
test_busline			pass		
test_checkResultsNum			pass		
test_checkaoi			pass		
test_checkfirstdata			pass		
test_checkfold			pass		
test_checkshangquan			pass		
test_checktendatas			pass		
test_express			pass		
test_kfc			pass		
test_routesearch			pass		
test_skip			pass		
test_skip2			pass		
test_spots			pass		
Total	**13**	**13**	**0**	**0**	

Part 03

功能自动化工具在搜索的应用实践

接口服务持续集成-pipeline

- PIPELINE交付流水线

	占用机器	编译&打包	部署&启动	功能回归	性能测试	Declarative: Post Actions	释放机器
	8s	1min 50s	3min 7s	8min 18s	1h 5min	33ms	10s
	10s	228ms	178ms	128ms	10h 1min	42ms	18s
	6s	439ms	4min 33s	15min 13s	166ms	31ms	8s
	9s	1s	4min 38s	15min 13s	169ms	32ms	8s
	5s	16min 17s	6min 43s	15min 13s	70ms	31ms	7s
	8s	3s	4min 29s	15min 13s failed	14ms	33ms	10s
	6s	664ms	5min 28s	15min 13s	71ms	32ms	9s
	9s	3min 49s failed	40ms	15ms	14ms	34ms	10s

接口服务持续集成-邮件

- 邮件通知执行结果

线上监控

- SHELL+crontab

```
function start_nlp_history()
{
    local server_inner='Nlp_history'
    local hostport="2[illegible]3"
    local allhost='zzdt=[illegible]'
    local shbt="[illegible].8"
    local shyc2="[illegible]3"
    local zzzc="[illegible]72"
    local zzdt="[illegible]61"
    local mailto_title="历史词接口监控-fromQAmonitor"
    local mailto_list="[illegible]360.cn"
    local cname="[illegible]"
    #local mailto_list="[illegible]360.cn"
    optionhost=`eval echo '$'"${hosts}"`
    if [[ $hosts != "all" ]];then
        main $hostport $server_inner $optionhost  $mailto_title $mailto_list $cname
    else
        main $hostport $server_inner ${allhost} $mailto_title $mailto_list $cname
    fi

}
```

线上监控-邮件和公众号通知

2019/9/2 (周一) 11:31

apitesti[illegible].360.cn

翻译垂搜接口监控

收件人 [illegible]

translate_englishtochinese_1567395013_error.csv
3 KB

失败时间	机房	ip/hostname	接口	query	失败原因
2019-09-02 11:30:16	zz[illegible]c	10.1[illegible].9[illegible].162	英译中接口	how are you	翻译 how are you 返回结果不符合预期|预期为你好吗实际为你好
2019-09-02 11:30:20	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	how are you	接口返回为空 status=|responsetime=
2019-09-02 11:30:20	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	nothing is diffcult	接口返回为空 status=|responsetime=
2019-09-02 11:30:20	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	It cost me a $12 taxi ride	接口返回为空 status=|responsetime=
2019-09-02 11:30:20	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	hello	接口返回为空 status=|responsetime=
2019-09-02 11:30:20	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	apple	接口返回为空 status=|responsetime=
2019-09-02 11:30:52	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	nothing is diffcult	接口返回为空 status=|responsetime=
2019-09-02 11:30:52	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	hello	接口返回为空 status=|responsetime=
2019-09-02 11:30:52	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	apple	接口返回为空 status=|responsetime=
2019-09-02 11:30:52	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	It cost me a $12 taxi ride	接口返回为空 status=|responsetime=
2019-09-02 11:30:52	zz[illegible]c	10.[illegible].[illegible].150	英译中接口	how are you	接口返回为空 status=|responsetime=

线上监控-邮件和公众号通知

SOQA消息

【视频接口监控-QA】
失败时间：2019-09-27 07:31:49
机房：zzdt:
接口：影视站短视频接口监控
query：马子跃现况
失败原因：影视站短视频result结果返回为空返回类型为<type list>|responsetime=91ms
接口负责人：@

【实体接口监控-fromQAmonitor】
失败时间：2019-09-27 10:25:09
机房：shbt:
接口：实体接口监控
query：
墟%7C辰东%
失败原因：result结果为空或者结果不正确|response_time=115 ms|type=小说-阅读|负责人:

【实体接口监控-fromQAmonitor】
失败时间：2019-09-27 10:25:43
机房：shbt:
接口：实体接口监控
query：query=末代皇帝
失败原因：type=影视-电影|不包含tagid:picture|response_time=161 ms|负责人:

接口线上监控-扩展功能-query白名单

- 选择性屏蔽监控query，避免解决问题时重复报警问题；---query加白名单功能
- 读取数据时，从列表中剔除白名单中的数据，不予监控；

接口线上监控-扩展功能-按百分比报错

- 如何避免因为数据正常变动而引起的误报？Config.ini文件中配置

```
20 info_log_file_path = %(newest_log_dir)s/%(section_name)s.csv
21 expected_pass_rate = 100.0
22
23 [test_checkfirstdata]
24 comment = 检查第一条结果正确性
25 section_name = test_checkfirstdata
26 data_file_path = %(data_dir)s/%(section_name)s.txt
27 info_log_file_path = %(newest_log_dir)s/%(section_name)s.csv
28 expected_pass_rate = 100.0
29
30 [test_checktendatas]
31 comment = 检查前十条结果正确性
32 section_name = test_checktendatas
33 data_file_path = %(data_dir)s/%(section_name)s.txt
34 info_log_file_path = %(newest_log_dir)s/%(section_name)s.csv
35 expected_pass_rate = 100.0
36
37 [test_express]
38 comment = 快递检测
39 section_name = test_express
40 data_file_path = %(data_dir)s/%(section_name)s.txt
41 info_log_file_path = %(newest_log_dir)s/%(section_name)s.csv
42 expected_pass_rate = 100.0
```

接口线上监控-如何避免污染线上数据

- 添加接口访问标签

gasucs/query_rec/?src=qa_test&ret_type=json&req=kw

Part 04

效果量化展示

主要应用效果

◆ 保障所有垂搜接口的功能自动化测试工作

- 覆盖十多个业务共72个接口，用例8000余条
- 嵌入多条持续集成流程中，服务端测试无须QA介入全自动化实现
- 赋能，满足多个业务的接口单独调用功能，用于开发人员本地调试接口

◆ 监控线上所有机房接口服务

- 多次发现线上问题，如第三方服务问题、环境问题、数据更新问题等等

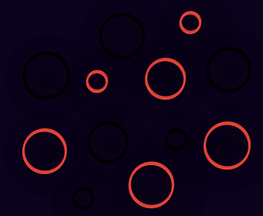
监控效果量化输出

◆ 监控问题分布图

结束语

- 有所启发，根据各自业务形态，解决业务痛点

脱离业务使用场景的测试工具都是耍流氓



Thanks