基于 python+Testlink+Jenkins 实现的接口自动化测试框架

by:授客 QQ：1033553122

博客：http://blog.sina.com.cn/ishouke
欢迎加入软件性能测试交流 QQ 群：7156436

目录

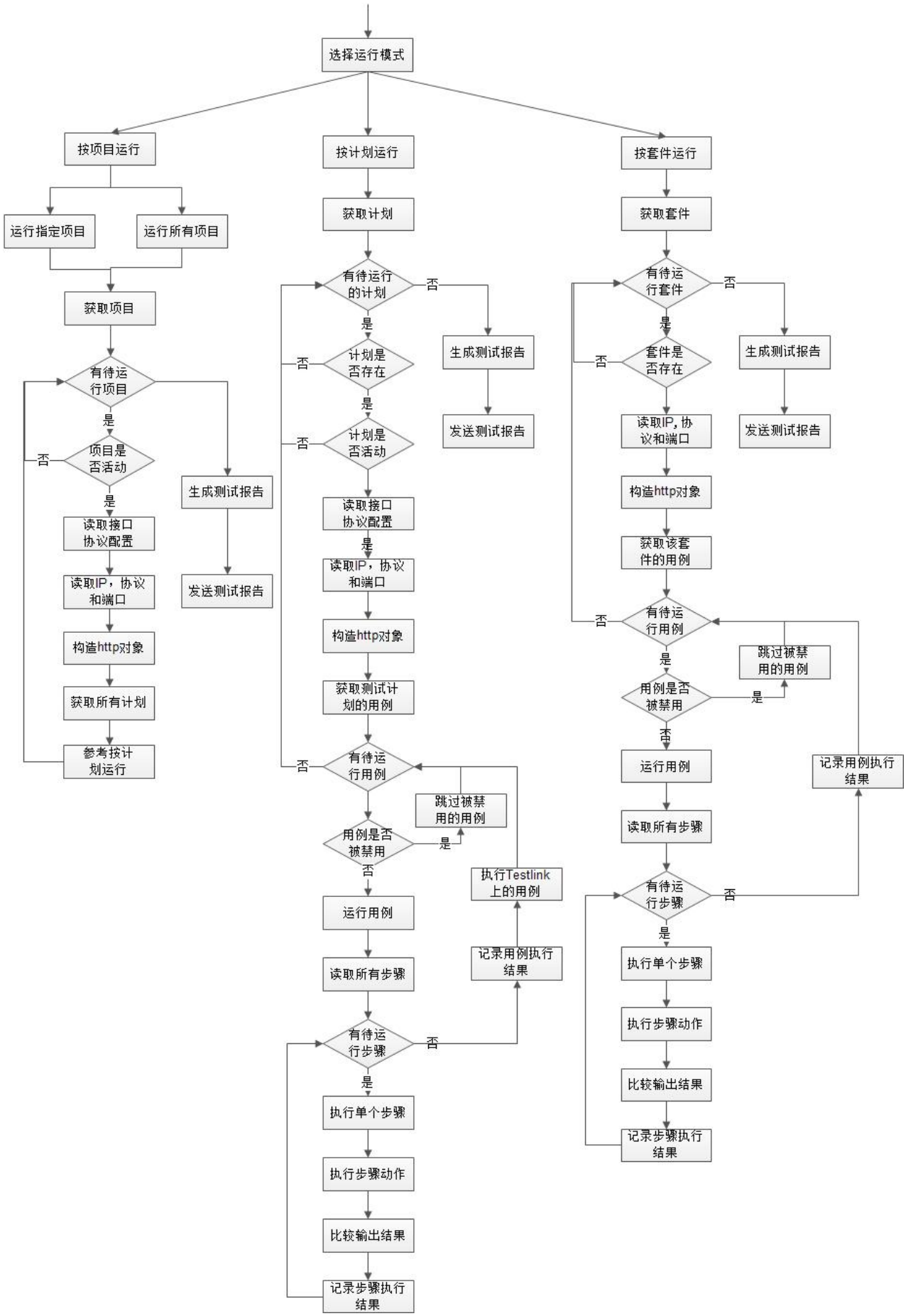
1、 开发环境

win7
PyCharm 4.0.5
python 3.3.2

testlink-1.9.14
下载地址：http://pan.baidu.com/s/1c16H50O
安装教程：暂略

TestLink-API-Python-client-master
下载地址：http://pan.baidu.com/s/1pLrcunT

2、 主要功能逻辑介绍


选择运行模式
按项目运行
运行指定项目
运行所有项目
获取项目
有待运行项目
是
否
项目是否活动
是
否
读取接口协议配置
读取IP，协议和端口
构造http对象
获取所有计划
参考按计划运行
生成测试报告
发送测试报告
按计划运行
获取计划
有待运行的计划
是
否
计划是否存在
是
否
计划是否活动
是
否
读取接口协议配置
是
读取IP，协议和端口
构造http对象
获取测试计划的用例
有待运行用例
否
跳过被禁用的用例
用例是否被禁用
是
否
运行用例
读取所有步骤
有待运行步骤
否
是
执行单个步骤
执行步骤动作
比较输出结果
记录步骤执行结果
记录用例执行结果
执行Testlink上的用例
生成测试报告
发送测试报告
按套件运行
获取套件
有待运行套件
是
否
套件是否存在
是
否
读取IP，协议和端口
构造http对象
获取该套件的用例
有待运行用例
否
是
用例是否被禁用
是
否
跳过被禁用的用例
运行用例
读取所有步骤
有待运行步骤
否
是
执行单个步骤
执行步骤动作
比较输出结果
记录步骤执行结果
记录用例执行结果
生成测试报告
发送测试报告

3、 框架功能简介

1、框架集成了 Testlink,可使用 Testlink 灵活对测试项目，测试计划，测试用例进行管理

2、可通过配置文件灵活配置运行模式，支持按测试项目、测试计划、测试套件批量运行执行用例

3、支持 HTTPS，HTTP，WebService 协议，支持 POST，GET 方法，支持 JSON，非 JSON 数据格式的请求，支持多种形式的数据校验

4、可自动生成 HTML 可视化接口测试报告

5、可根据配置在测试完成后，自动发送测试报告邮件

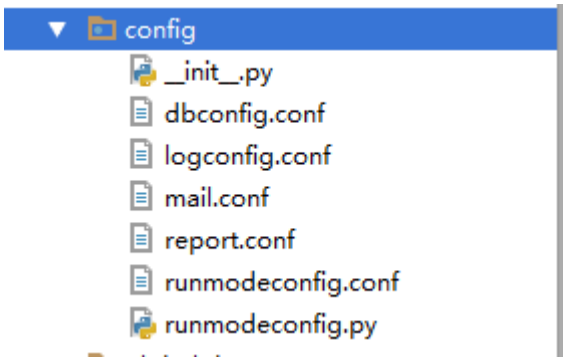
6、支持文件、控制的日志打印，可分别控制开关

7、支持模块化开发

8、可集成 Jenkins 自动运行脚本
参考文章：为 Jenkins 添加 Windows Slave 远程执行 python 项目脚本

4、 框架模块详细介绍

a) config



dbconfig.conf：包含测试数据库，应用数据库的配置信息

logconfig.conf：包含日志配置信息，具体如下：

logconfig.conf

```
[LOGGING]
log_file = F:\project\interface_project\logs\log.txt        日志文件所在路径
max_bytes_each = 51200        单个日志文件大小
backup_count = 10        同一时刻，可存留的日志文件数
fmt = |(asctime)s |(filename)s[line: |(lineno)d] |(levelname)s: |(message)s
logger_name = test_logger        日志打印器名称
log_level_in_console = 10        日志格式
log_level_in_logfile = 20        日志级别
console_log_on = 1
logfile_log_on = 1        日志打印开关

[README]
log_level = '日志级别: CRITICAL = 50 ERROR = 40 WARNING = 30 INFO = 20 DEBUG = 10 NOTSET = 0'
log_on = 'console_log_on  = 1 开启控制台日志，0则关闭，logfile_log_on = 1 开启文件日志， 0则关闭'
```

mail.conf：包含邮件发送配置信息，如下，

mail.conf

```
[SMTP]
login_user = laiyuhenshuai@163.com        邮件发送人帐号密码
login_pwd = xxxxhuozhe
from_addr =  laiyuhenshuai@163.com
to_addrs = ['mrxxx@163.com','1033553122@qq.com']        要发送的目标邮箱，
host = smtp.163.com                                     支持多方发送
port = 25
```

注：不同类型的邮箱(发件人邮箱)，需要修改配置文件为对应的 host 和端口

smtp.163.com:25

smtp.qq.com:465

report.conf：包含测试报告文件配置信息，如下

report.conf

```
[REPORT]
dir_of_report = F:\\project\\report\\
report_name = test_report.html

[README]
dir_of_path = '不能加引号,'D:\\tset\\tkise\\,  r'D:\tset\tkise\
report_name = '不能加引号'
```

测试报告所在路径

测试报告文件名

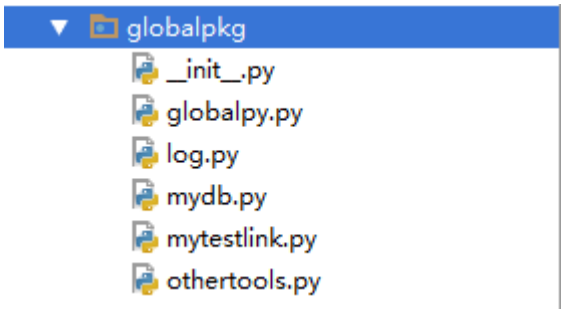
runmodeconfig.conf：包含运行模式配置信息

runmodeconfig.conf

```
[RUNMODE]
runmode = 1

[PROJECTS]
project_mode = 2
projects = ['项目1','项目2', '项目3', '项目4']

[PLANS]
project = 项目1
plans = ['项目1-测试计划1', '项目1-测试计划2', '项目1-测试计划3', '项目1-测试计划4', '项目-测试计划4', '项目1-测试计划5']

[TESTSUITES]
testsuites = [3, 11, 31, 35,38,43,48, 56, 65, 69]

[README]
runmode = 'runmode: 1 - 按项目运行 2 - 按计划运行 3 - 按套件运行
testsuites = ['套件1id', '套件2id', '...','套件Nid']
plans = '项目及项目关联的测试计划'
projects = '如果project_mode=2,那么需要配置需要运行的项目，如果project_mode配置为1，则运行所有项目'
```

runmodeconfig.py：日志配置类

b) globalpkg



log.py：实现日志打印类

mydb.py：实现数据库类，封装数据库相关操作

mytestlink.py：主要用于获取 testlink 连接实例

othertools.py：实现其它通用功能，比如数据转换，批量创建目录等

globalpy.py：主要提供全局变量，全局实例等

globalpy.py ×

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-

__author__ = 'shouke'

import ...


logger.info('正在初始化数据库[名称: TESTDB]对象')
testdb = MyDB('./config/dbconfig.conf', 'TESTDB')

logger.info('正在获取testlink')
mytestlink = TestLink().get_testlink()

other_tools = OtherTools()

executed_history_id = time.strftime('%Y%m%d%H%M%S', time.localtime())  # 流水记录编号
# testcase_report_tb = 'testcase_report_tb' + str(executed_history_id)
# case_step_report_tb = 'case_step_report_tb' + str(executed_history_id)
testcase_report_tb = 'testcase_report_tb'
case_step_report_tb = 'case_step_report_tb'
```

可根据需要，在这添加应用db

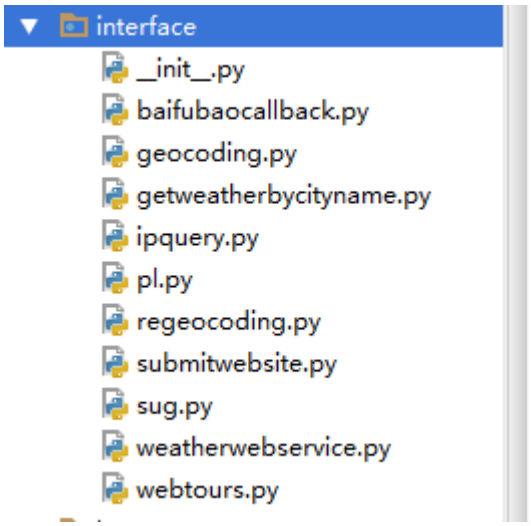
可根据实际需要，未每次测试运行新建测试用例及步骤报表，也可以设置为两张固定的表

c) **logs 及 testreport**

可分别用于存放日志文件，测试报告

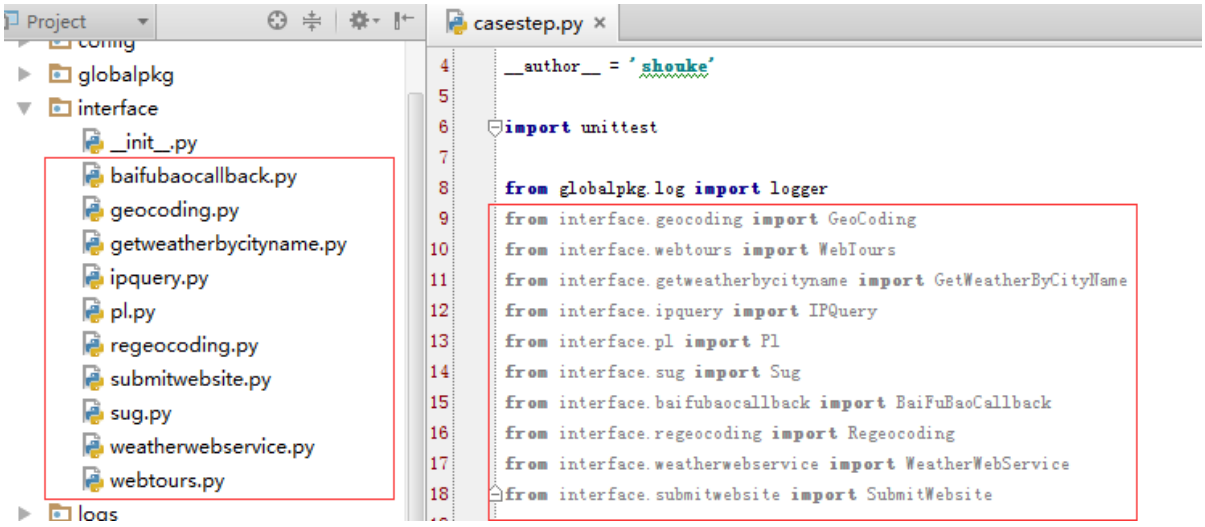
d) **interface**

封装接口测试方法类



说明：可根据需要，每个接口对应一个模块，对应一个类；也可以多个接口对应一个模块，对应一个类

需要注意的是，这里添加的模块及类，需要在 casestep.py 中导入



目前框架中的模块给出了一些案例，如下：

getweatherbycityname.py

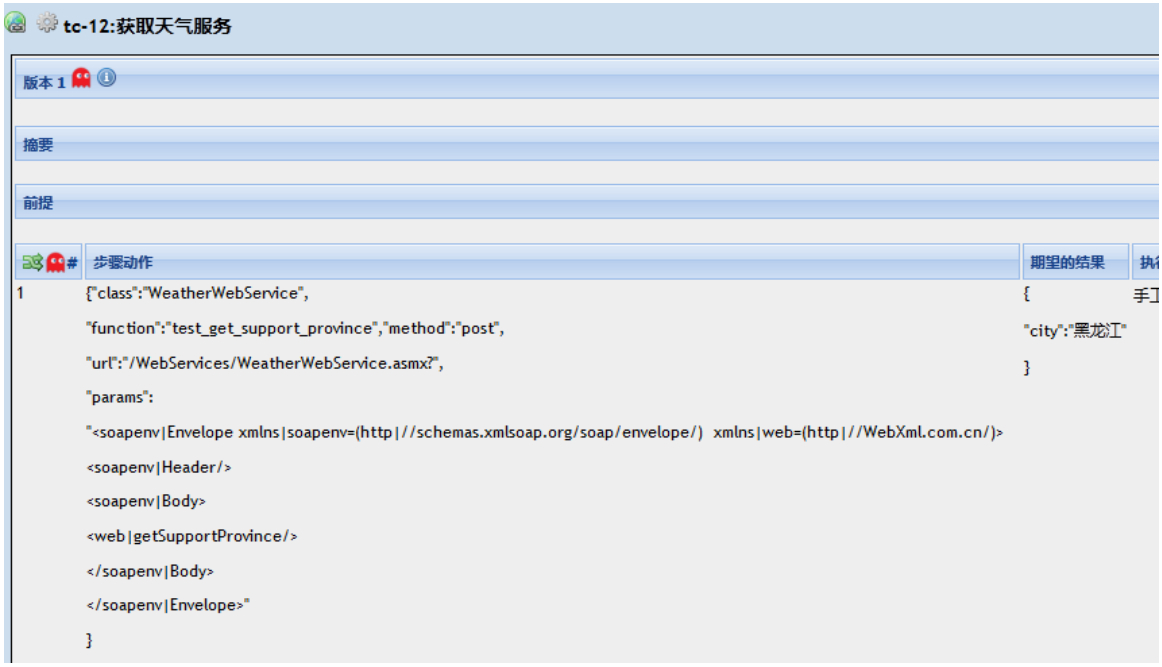
```
__author__ = 'shouke'

import ...

class GetWeatherByCityName(MyUnittestTestCase):
    def setUp(self):
        pass

    # 测试天气查询接口
    def test_get_weather_by_city_name(self):
        """演示接口返回数据为xml（webservice）格式的数据处理"""

        self.params = (self.params)[0]
        self.params = urllib.parse.urlencode(self.params)

        logger.info('正在发起GET请求...')
        response = self.http.get(self.url, self.params)
        response = response.decode('utf-8')

        print(response)
        logger.info('正在解析返回结果')
        root = ET.fromstring(response)

        # 断言
        self.assertEqual(root[2].text,  self.expected_result['city'], msg='直辖市不等于上海')
        self.assertEqual(root[2].text,  self.expected_result['citycode'], msg='城市代码不等于58367')


    def tearDown(self):
        pass
```

说明：一个用例步骤对应这里的一个test_xxx方法

weatherwebservice.py

```
class WeatherWebService(MyUnittestTestCase):
    def setUp(self):
        pass

    #测试获取支持的省市
    def test_get_support_province(self):
        '''演示Body为XML格式数据的POST请求'''
        header = {'Content-Type':'text/xml','charset':'utf-8'}
        self.http.set_header(header)

        # 参数处理
        self.params = self.params.replace('|',':')
        self.params = self.params.replace('(','"')
        self.params = self.params.replace(')','"')

        logger.info('正在发起POST请求...')

        self.params = self.params.encode(encoding='utf-8')

        response = self.http.post(self.url,  self.params)
        response = response.decode('utf-8')

        logger.info('正在解析返回结果:%s' % response)
        root = ET.fromstring(response)

        city_heilongjiang = root[0][0][0][2]
        # 断言
        self.assertEqual(city_heilongjiang.text,  self.expected_result['city'], msg='支持天气服务的城市少了"黑龙江"')

    def tearDown(self):
        pass
```

对应用例配置：



webtours.py

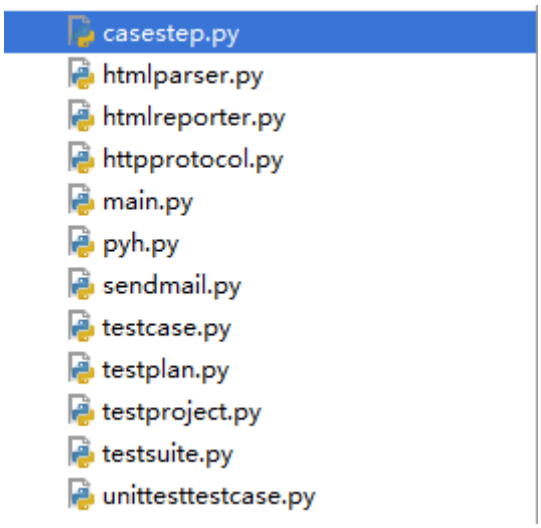
```
class WebTours(MyUnittestTestCase):
    def setUp(self):
        pass

    # 测试访问WebTours首页
    def test_visit_webtours(self):
        # 根据被测接口的实际情况，合理的添加HTTP头
        # header = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
        #     'User-Agent':'Mozilla/5.0 (Windows NT 6.1; rv:29.0) Gecko/20100101 Firefox/29.0'
        #     }
        logger.info('正在发起GET请求...')
        response = self.http.get(self.url,  (self.params))

        logger.info('正在解析返回结果')

        # 解析HTML文档
        parser = MyHTMLParser(strict = False)
        parser.feed(response)

        # 比较结果
        starttag_data = parser.get_starttag_data()
        i = 0
        for data_list in starttag_data:
            if data_list[0] == 'title' and data_list[1] == 'Web Tours':
                i = i + 1

        self.assertNotEqual(str(i), self.expected_result['result'], msg='访问WebTours失败')

        # 如果有需要，连接数据库，读取数据库相关值，用于和接口请求返回结果做比较
```

演示返回结果为HTML文档时的数据处理

更多案例烦自行查阅模块

e) 其它模块

如下，顾名思义



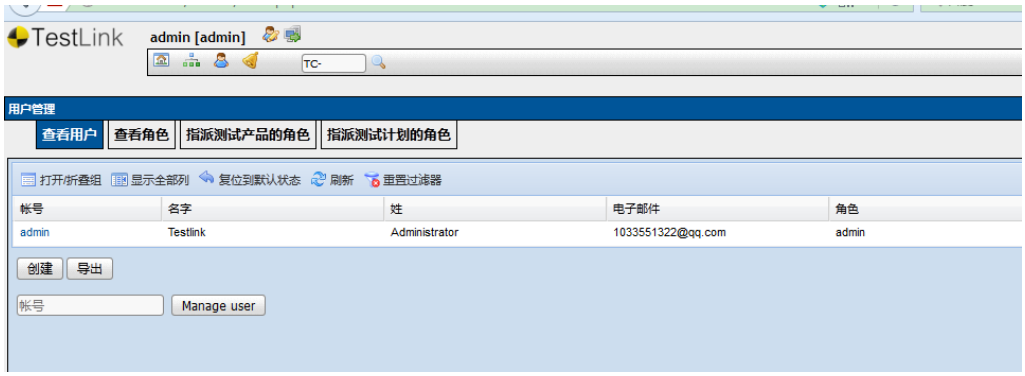
5、 Testlink 相关配置与用例管理

为了批量设置接口 ip，端口(主要是这两个)，协议信息(仅用于展示)，需要对项目，计划，套件等必要的配置，

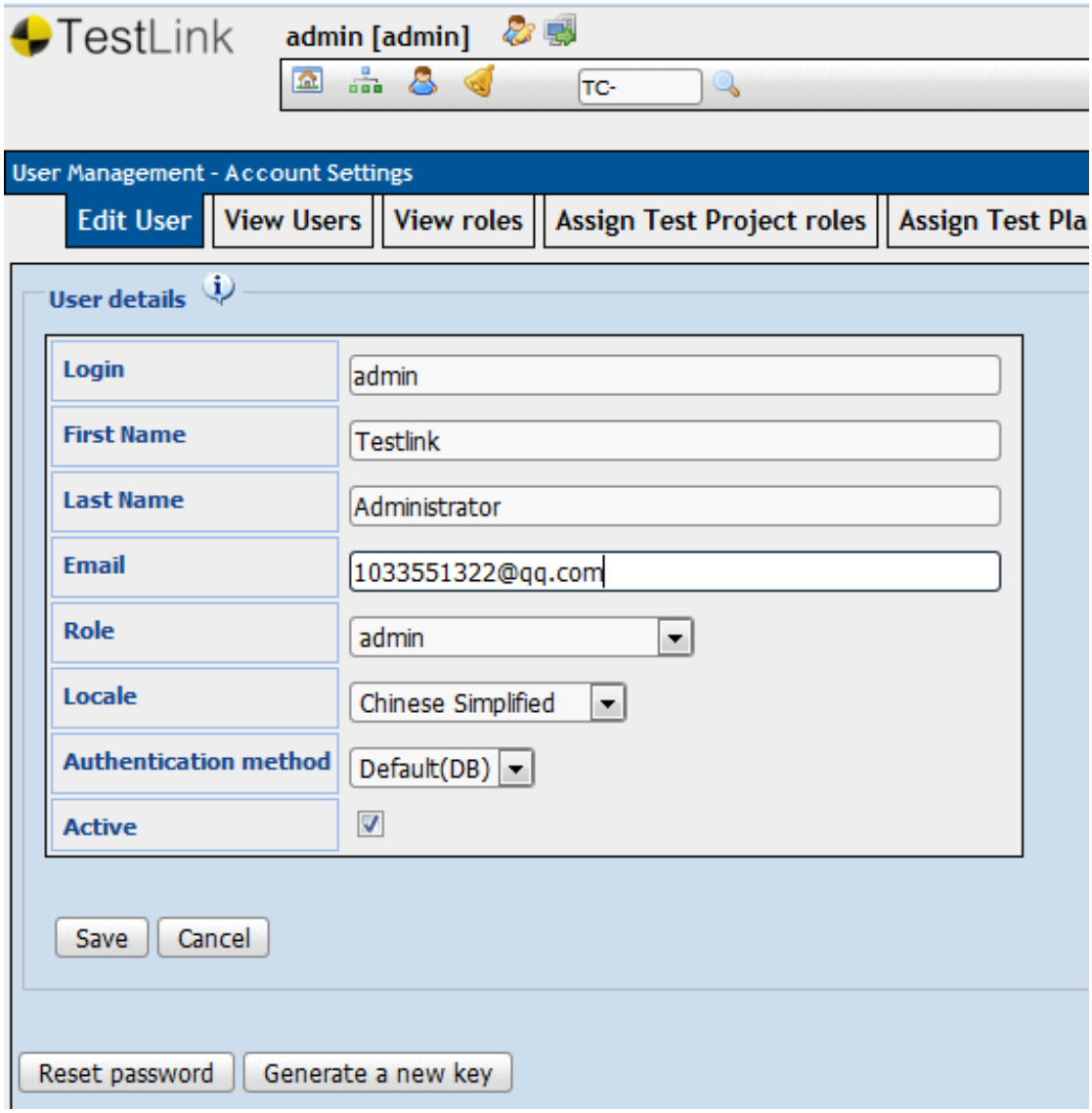
以及客户端环境变量配置

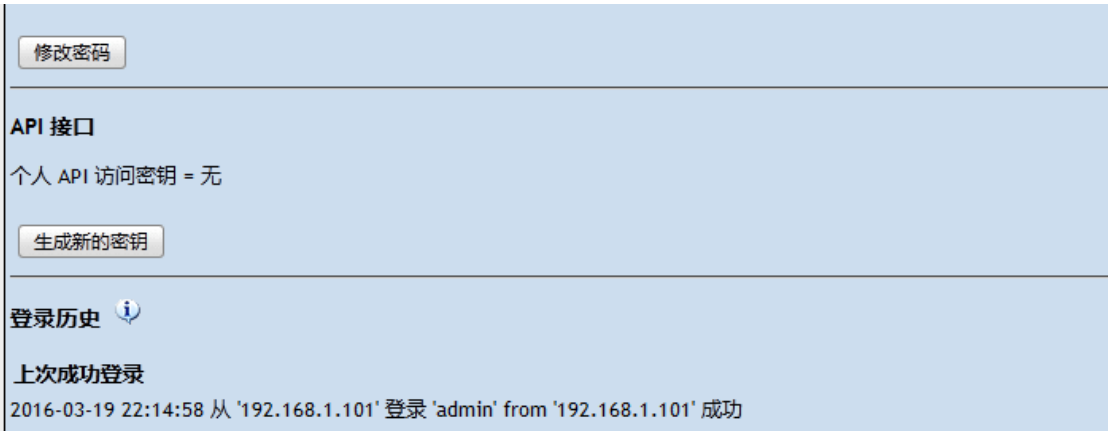
a） 配置

如下，登陆 Testlink，进入用户管理-查看用户，如下

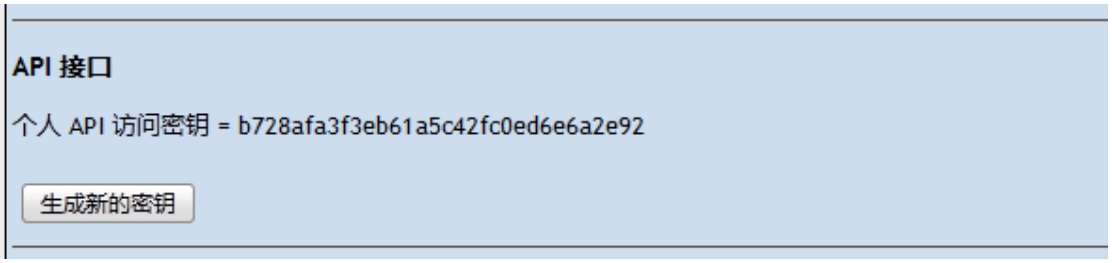


点击目标用户(例中为 admin)，打开如下界面



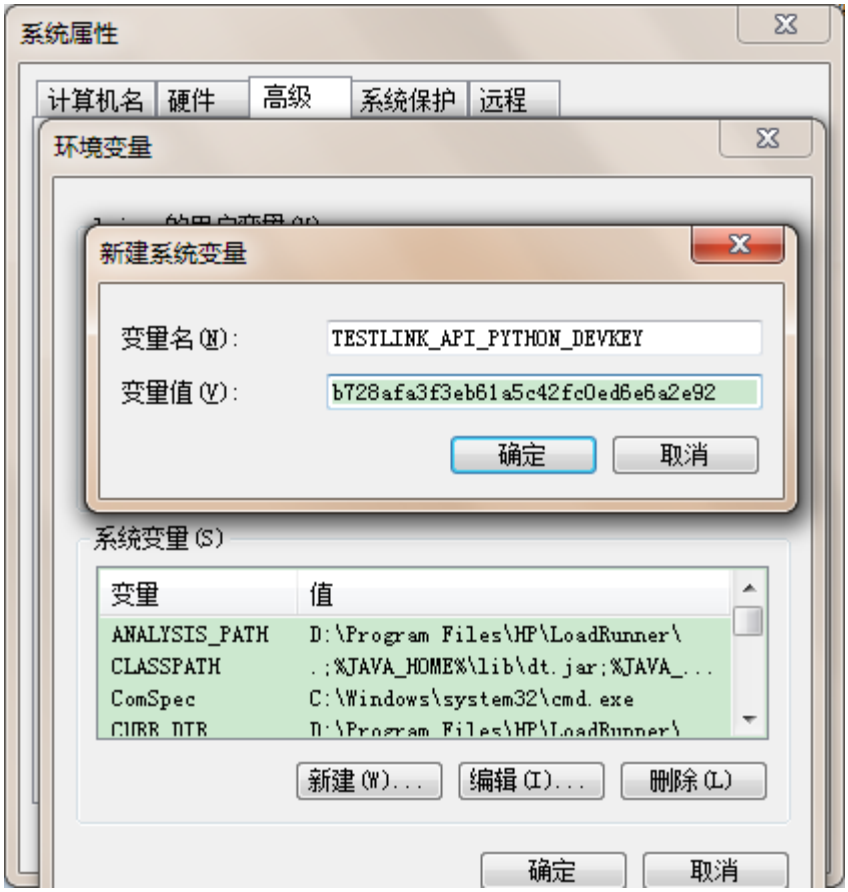


点击生成新的密钥，如下

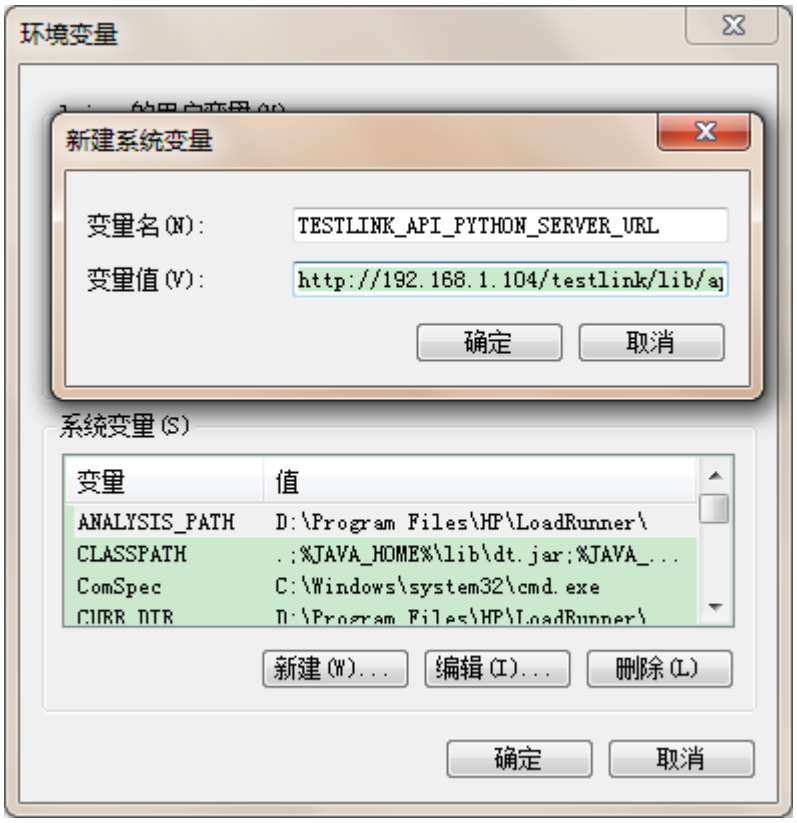


在运行 python 脚本端进行环境变量的配置，如下：

1、新建系统环境变量“TESTLINK_API_PYTHON_DEVKEY”，变量值为上述秘钥



2、新建“TESTLINK_API_PYTHON_SERVER_URL”系统环境变量，变量值为“http://{host}/testlink/lib/api/xmlrpc/v1/xmlrpc.php”,其中 host 为 testlink 的访问地址

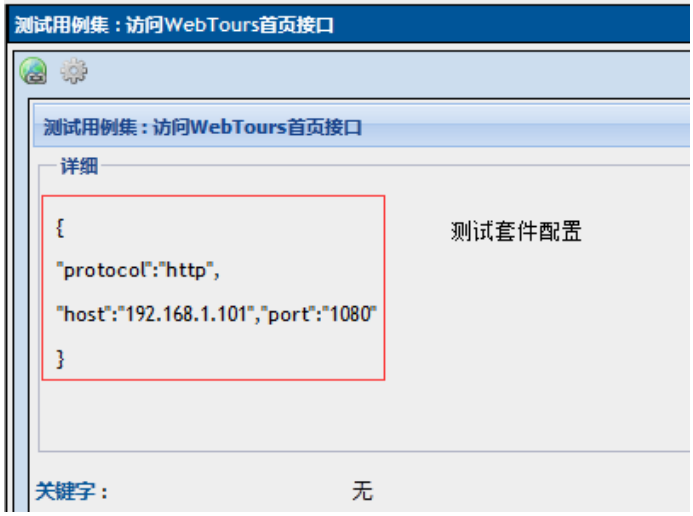


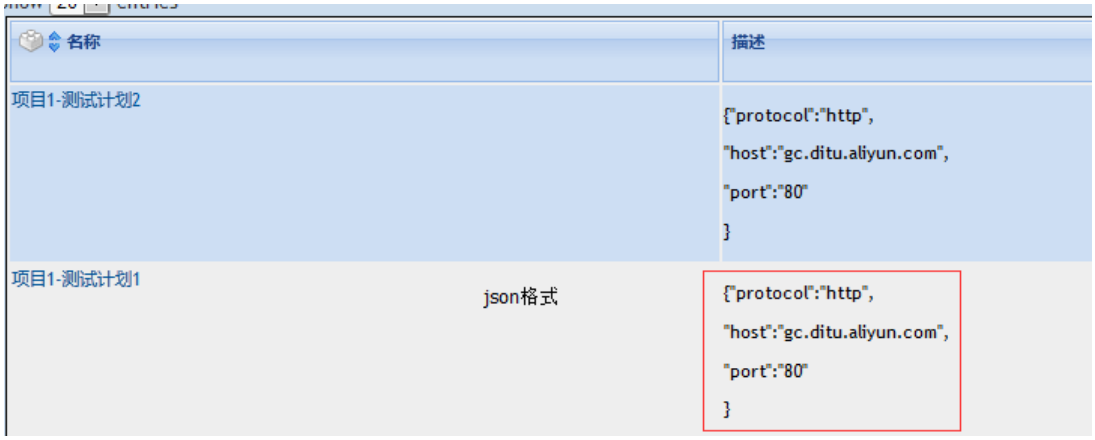
测试是否生效：

```
C:\Users\laiyu>python
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32
tel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import testlink
>>> tls = testlink.TestLinkHelper().connect(testlink.TestlinkAPIClient)
>>> tls.testLinkVersion()
'1.9.14'
```

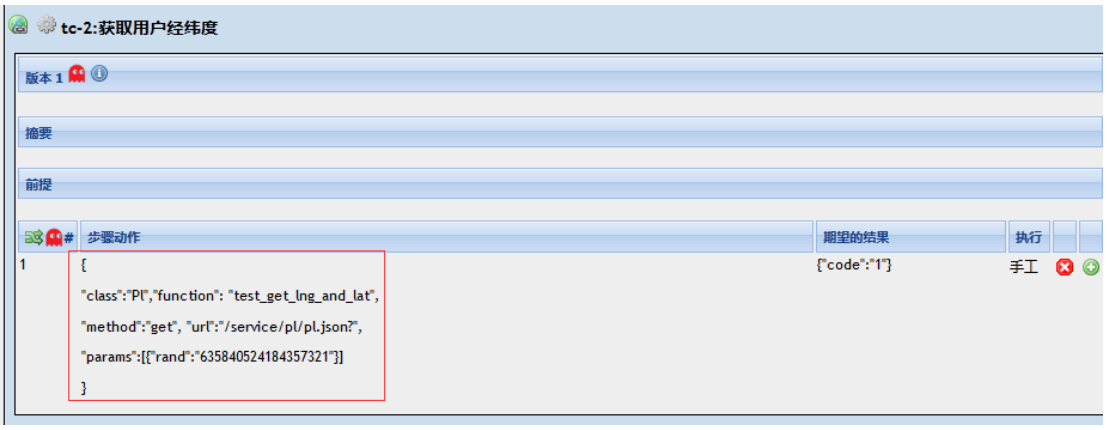
项目，计划，套件等相关配置

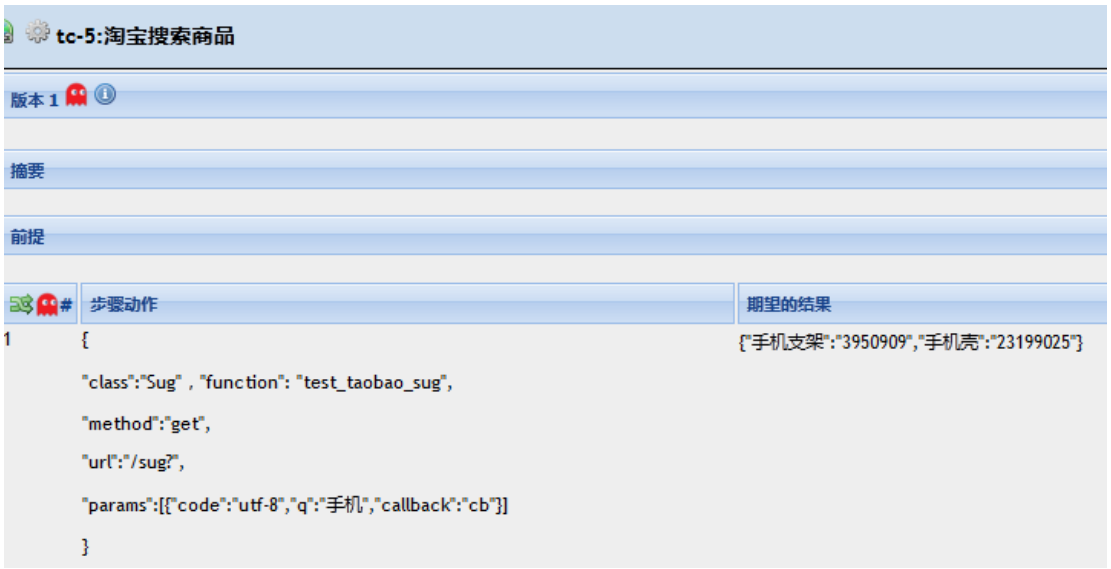

测试产品：项目1
产品名称
项目1
说明
{
"protocol":"http",
"host":"gc.ditu.aliyun.com",
"port":80
}
json格式

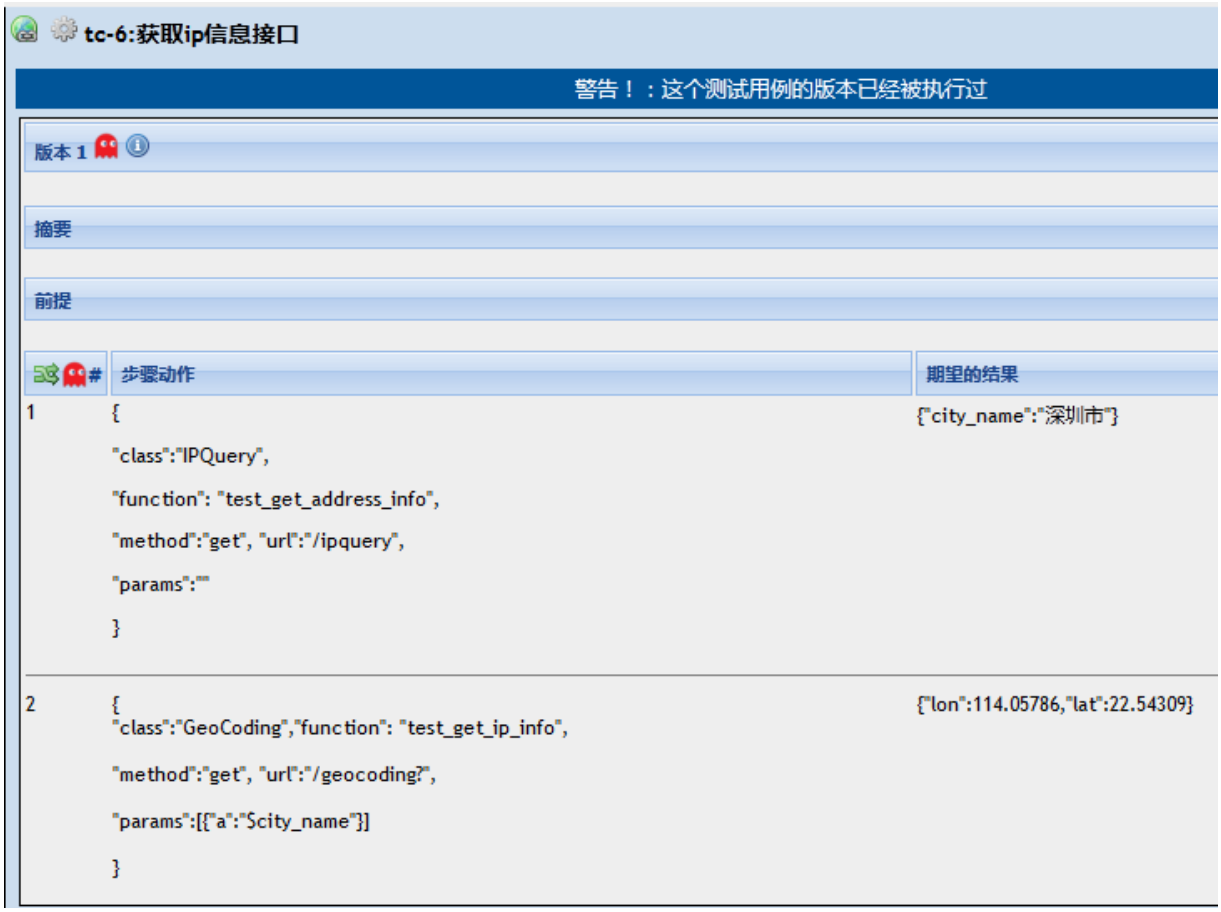

测试用例集：访问WebTours首页接口
测试用例集：访问WebTours首页接口
详细
{
"protocol":"http",
"host":"192.168.1.101","port":"1080"
}
测试套件配置
关键字：
无

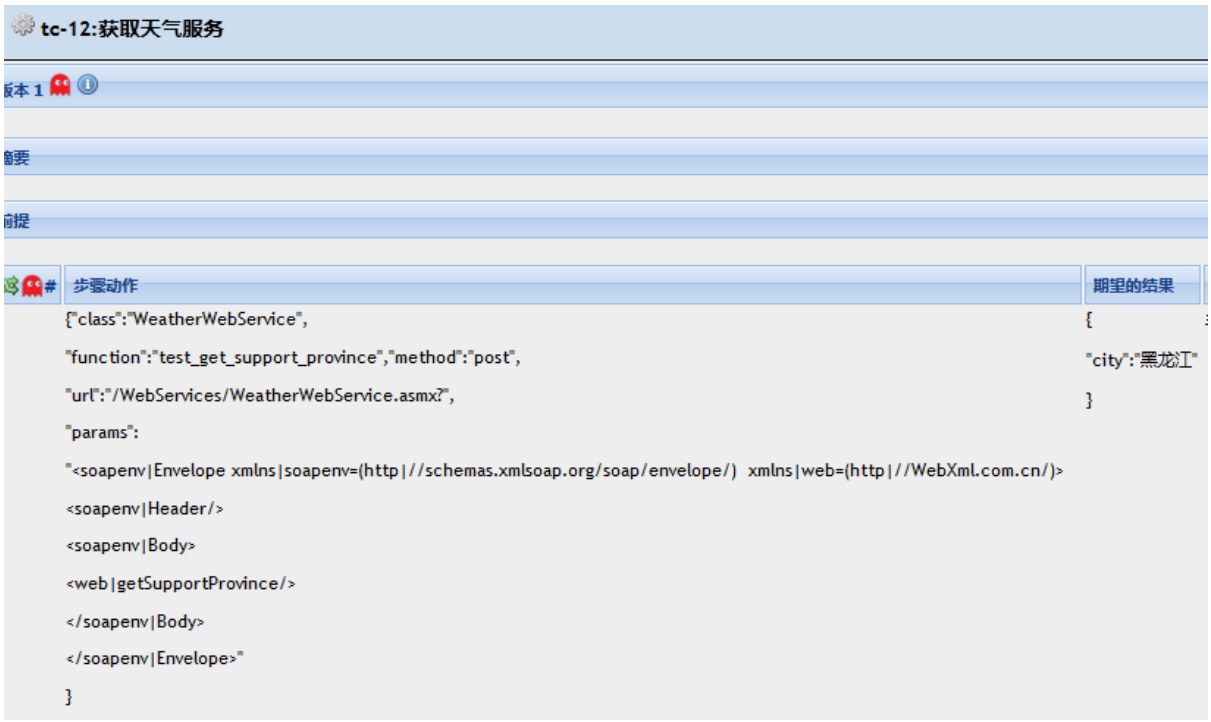

名称
描述
项目1-测试计划2
{"protocol":"http",
"host":"gc.ditu.aliyun.com",
"port":"80"
}
项目1-测试计划1
json格式
{"protocol":"http",
"host":"gc.ditu.aliyun.com",
"port":"80"
}

b） 用例管理


tc-2:获取用户经纬度
版本 1
摘要
前提
步骤动作
期望的结果
执行
1
{
"class":"Pl","function": "test_get_lng_and_lat",
"method":"get", "url":"/service/pl/pl.json?",
"params":[{"rand":"6358405241843573 21"}]
}
{"code":"1"}
手工


tc-5:淘宝搜索商品
版本 1
摘要
前提
步骤动作
期望的结果
1
{
"class":"Sug" , "function": "test_taobao_sug",
"method":"get",
"url":"/sug?",
"params":[{"code":"utf-8","q":"手机","callback":"cb"}]
}
{"手机支架":"3950909","手机壳":"23199025"}


tc-6:获取ip信息接口
警告！：这个测试用例的版本已经被执行过
版本 1
摘要
前提
步骤动作 期望的结果
1 { {"city_name":"深圳市"}
"class":"IPQuery",
"function": "test_get_address_info",
"method":"get", "url":"/ipquery",
"params":""
}
2 { {"lon":114.05786,"lat":22.54309}
"class":"GeoCoding","function": "test_get_ip_info",
"method":"get", "url":"/geocoding?",
"params":[{"a":"$city_name"}]
}


tc-12:获取天气服务
# 步骤动作 期望的结果
{"class":"WeatherWebService", {
"function":"test_get_support_province","method":"post", "city":"黑龙江"
"url":"/WebServices/WeatherWebService.asmx?", }
"params":
"<soapenv|Envelope xmlns|soapenv=(http|//schemas.xmlsoap.org/soap/envelope/) xmlns|web=(http|//WebXml.com.cn/)>
<soapenv|Header/>
<soapenv|Body>
<web|getSupportProvince/>
</soapenv|Body>
</soapenv|Envelope>"
}

6、 运行结果

见源码附件

7、 源码下载

下载地址：http://pan.baidu.com/s/1c2Av9FM

下载后解压，用 pycharm 导入项目即可

8、 说明

时间有限，精力有限，暂且就到这吧，有需要的可以自己扩展、修改框架。

注：目前还存在个 bug，测试报告中，类似 xml 格式数据没显示出来，有兴趣的烦先自己解决下。