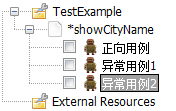
搭建思路

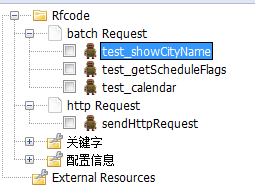
框架采用robot和python实现，因为robot中的复杂逻辑实现起来比较繁琐，故选择用python实现，然后以外部库的形式导入robot中使用。测试用例数据保存在excel中。  
使用过robot的人了解，robot中测试维度划分为测试套件（Test Suite）和测试用例（Test Case），一个Suite为一组Case的集合，每个Case对应为我们手工执行测试时的Case。

假设测试一个路径为/areaplug/showCityName的http接口，常规方法是在robot中新建一个showCityName的Suite，其下包含测试该http接口的用例集，如下图所示：



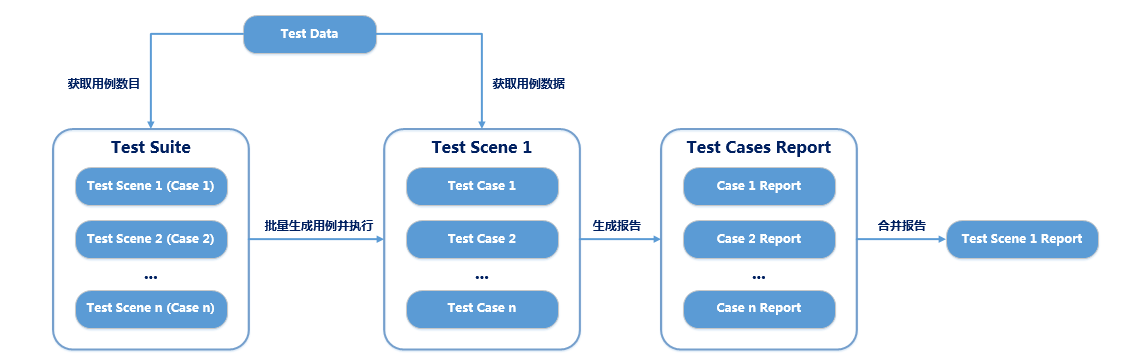
showCityName Test Suite

倘若测试该接口有20个异常用例，则建立20条相应的test case。但是，对于测试http接口来讲，以上用例无非就是请求参数和响应不一样而已，发送请求的逻辑是一模一样的。所以，这20条test case其实用一条test case就能实现了，在这1条case中分别遍历读取20个异常用例的测试数据执行测试就ok了。所以最后构造的suite和case如下：



接口case

图中，batch\_Request为测试套件，其下的每个robot的test case对应一个http接口测试场景，比如测试路径为/areaplug/showCityName的http接口，该接口的所有正向和异常用例均在test\_showCityName中实现，在test\_showCityName中读取测试数据文件，获取该接口的测试用例数目，遍历每一条测试用例数据，调用http\_Request下的sendHttpRequest发送http请求。其实，这里的test\_showCityName就相当于test suite了，而遍历测试数据文件中的每一行测试数据去调用sendHttpRequest时，就相当于生成了一条test case，这样就可以将一个接口的所有测试用例用robot的一条test case实现（实质是robot的一条test case相当于一个test suite，在这个robot的test case中动态生成n条test case）。整个流程如下图所示：

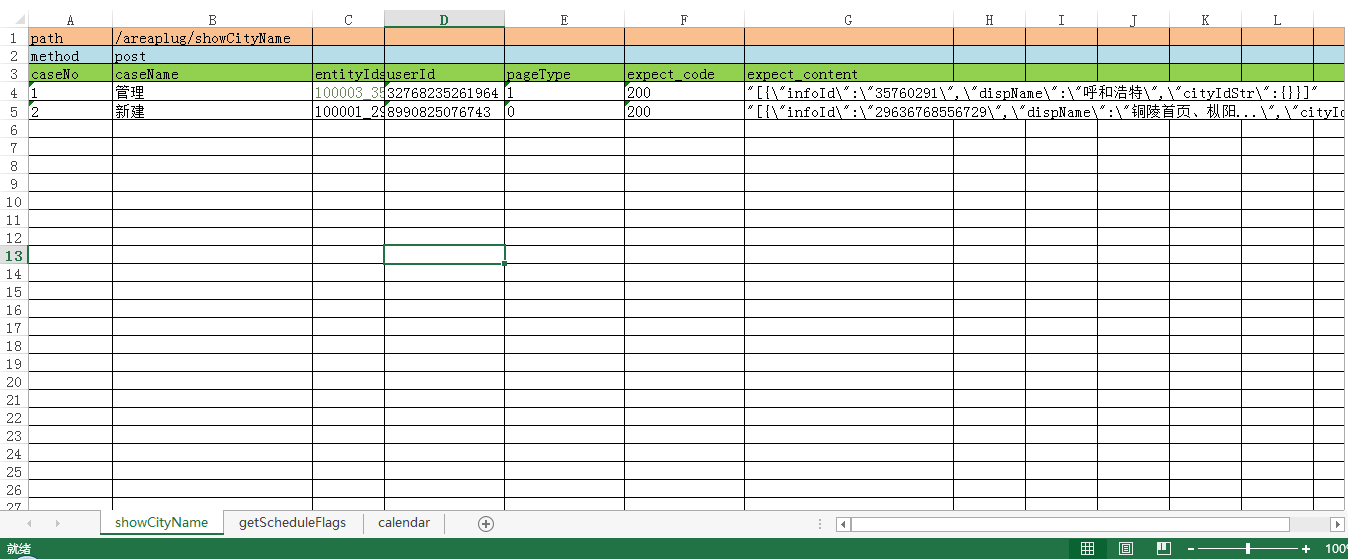


框架流程图

搭建

测试数据

测试数据保存在excel中，每一个sheet页对应一个测试场景，即一个http接口。该sheet也保存有测试该接口的所有测试用例数据以及接口路径和请求方法，如下图所示（这里仅仅是一个demo，实际回归测试时，会有大量的用例和数据）：



测试数据

测试框架

整个工程目录如下：

E:\LLF\_58TESTSUITES\JZ\_WEBINTERGRATION\ROBOT\_CODE

│ execPybot.bat

│

├─pycode

│ │ Common\_Excel.py

│ │ Common\_Excel.pyc

│ │ Common\_Exec.py

│ │ Common\_Exec.pyc

│ │ testHTTP.py

│ │ \_\_init\_\_.py

│ │

│ ├─.idea

│ │ │ misc.xml

│ │ │ modules.xml

│ │ │ pycode.iml

│ │ │ workspace.xml

│ │ │

│ │ └─inspectionProfiles

│ └─\_\_pycache\_\_

│ Common\_Excel.cpython-36.pyc

│ Common\_Exec.cpython-36.pyc

│ \_\_init\_\_.cpython-36.pyc

│

├─report

│ │ log.html

│ │ output.xml

│ │ report.html

│ │

│ └─TestCaseReport

│ ├─result\_calendar

│ │ log\_20180130195712.html

│ │ output\_20180130195712.xml

│ │ report\_20180130195712.html

│ │

│ ├─result\_getScheduleFlags

│ │ log\_20180130195710.html

│ │ output\_20180130195710.xml

│ │ report\_20180130195710.html

│ │

│ └─result\_showCityName

│ log\_20180130195707.html

│ output\_20180130195707.xml

│ report\_20180130195707.html

│

├─rfcode

│ │ batch\_Request.txt

│ │ http\_Request.txt

│ │ \_\_init\_\_.robot

│ │

│ ├─关键字

│ │ 关键字index.txt

│ │ 自定义关键字.txt

│ │

│ └─配置信息

│ config.txt

│ configIndex.txt

│ RequestHeaders.txt

│

└─testData

testData.xlsx

工程有4部分构成：

* pycode  
  由于robot中复杂逻辑的实现比较繁琐，所以将一些复杂逻辑直接用python代码实现，然后以外部库的形式导入robot中调用。共有2个文件:

Common\_Excel.py  
主要负责对测试数据excel文件的读取操作。

# coding: utf-8

import xlrd

def getTestData(testDataFile, testScene, host, caseNo):

'''

从excel中获取测试数据

:param testDataFile: 测试数据文件

:param testScene: 测试场景

:param host: 服务器主机

:param caseNo: 用例No

:param method: 请求方法

:return: url，用例No，用例名称，请求参数，预期返回码，预期响应内容

'''

caseNo = int(caseNo)

data = xlrd.open\_workbook(testDataFile)

table = data.sheet\_by\_name(testScene)

cols = table.ncols

resource\_path = table.cell(0, 1).value # 文件路径

url = "http://" + host + resource\_path # 访问的url

method = table.cell(1, 1).value # 请求方法

dict\_params = {}

for i in range(cols):

dict\_params[table.cell(2, i).value] = table.cell(caseNo+2, i).value

caseNo = dict\_params.pop("caseNo")

caseName = dict\_params.pop("caseName")

expectCode = dict\_params.pop("expect\_code")

expectCotent = dict\_params.pop("expect\_content")

testName = "TestCase" + caseNo + "\_" + caseName

return method, url, caseNo, testName, dict\_params, expectCode, expectCotent

def getTestCaseNum(testDataFile, testScene):

'''

获取testScene测试场景中的测试用例数

:param testDataFile: 测试数据文件

:param testScene: 测试场景

:return: 测试用例数

'''

data = xlrd.open\_workbook(testDataFile)

table = data.sheet\_by\_name(testScene)

rows = table.nrows

return rows-3

def getTestHttpMethod(testDataFile, testScene):

'''

获取testScene测试场景的请求方法

:param testDataFile: 测试数据文件

:param testScene: 测试场景

:return: 请求方法

'''

data = xlrd.open\_workbook(testDataFile)

table = data.sheet\_by\_name(testScene)

method = table.cell(1, 1).value # 请求方法

return method

Common\_Exec.py  
主要负责根据测试数据批量构造pybot命令来调用robot执行测试。

# coding: utf-8

import requests

import os

import time

def batch\_Call(robot\_testSuite, robot\_testCase, testScene, caseNum, testCaseReportPath, execTime):

'''

批量执行testScene测试场景下的用例

:param robot\_testSuite: robot testSuite路径

:param robot\_testCase: robot testCase路径

:param testScene: 测试场景

:param caseNum: 用例数

:param testCaseReportPath: 业务用例测试报告路径

:param execTime: 执行时间

:return:

'''

try:

for caseNo in range(caseNum):

testCase = ""

caseNo = caseNo + 1

testName = "testcase" + "\_" + str(caseNo)

output\_dir = "-d " + testCaseReportPath + "/result\_{0}".format(testScene) # 输出目录

output\_xml = "-o output\_{0}\_{1}.xml".format(testName, execTime)

output\_log = "-l log\_{0}\_{1}.html".format(testName, execTime)

output\_report = "-r report\_{0}\_{1}.html".format(testName, execTime)

variable = "-v caseNo:" + str(caseNo) + " -v testScene:" + testScene

testCase = "--test " + robot\_testCase

pybot\_cmd = "pybot " + output\_dir + " " + output\_xml + " " + output\_log + " " + output\_report + " " + variable + " " + " " + testCase + " " + robot\_testSuite

os.system(pybot\_cmd) # 执行pybot命令

return "done"

except Exception as e:

return "Error: " + str(e)

def send\_HttpRequest(url, data=None, headers=None, method=None):

'''

发送http请求

:param url: 请求的url

:param data: 请求数据

:param headers: 请求头

:param method: 请求方法

:return: 响应码，响应内容

'''

if method == "get":

response = requests.get(url, data, headers=headers)

if method == "post":

response = requests.post(url, data, headers=headers)

code = str(response.status\_code)

content = response.content.decode("utf-8") # 转码

return code, content

def cleanLogs(testScene, testCaseReportPath):

'''

删除硬盘中合并前的测试报告

:param testScene: 测试场景

:param testCaseReportPath: 业务用例测试报告路径

:return:

'''

testCaseReportPath = testCaseReportPath + "/result\_{0}".format(testScene)

report\_files = testCaseReportPath + "/report\_testcase\*"

xml\_files = testCaseReportPath + "/output\_testcase\*"

log\_files = testCaseReportPath + "/log\_testcase\*"

cmd = "del " + report\_files + " " + xml\_files + " " + log\_files # windows

cmd = cmd.replace("/", "\\")

print(cmd)

os.system(cmd)

def getCurtime():

'''

获取当前时间

:return: 当前时间

'''

return time.strftime("%Y%m%d%H%M%S", time.localtime(time.time()))

def mergeReport(testScene, testCaseReportPath, execTime):

'''

# 合并报告

:param testScene: 测试场景

:param testCaseReportPath: 业务用例测试报告路径

:param execTime: 执行时间

:return:

'''

try:

output\_dir = "-d " + testCaseReportPath + "/result\_{0}".format(testScene) # 输出目录

output\_xml = "-o output\_{0}.xml".format(execTime)

output\_log = "-l log\_{0}.html".format(execTime)

output\_report = "-r report\_{0}.html".format(execTime)

# 被合并的报告

merge\_report = testCaseReportPath + "/result\_{0}".format(testScene) + "/output\_testcase\_\*.xml"

name = "--name " + testScene

rebot\_cmd = r"rebot " + output\_dir + " " + output\_xml + " " + output\_log + " " + output\_report + " " + name + " " + merge\_report

os.system(rebot\_cmd) # 执行rebot命令

return "done"

except Exception as e:

return "Error: " + str(e)

* report  
  该目录用于存放测试报告。其中report目录下的robot测试报告为测试Suite的测试报告，而TestCaseReport下会根据不同的测试场景生成对应该场景名称的测试报告文件夹，其下会包含该测试场景下所有用例的合并报告（即excel中的每一条case会生成一个报告，最后会将这些cases的报告合并为一个报告，作为该测试场景即该http接口的测试报告）。
* rfcode  
  该目录下为robot的代码。

batch\_Request.txt  
batch\_Request下包含要测试的各http接口对应的测试场景（即robot的测试用例）。在各测试场景中会设置${testScene}变量，通过该变量去excel文件中对应的sheet页获取相应的测试数据。

\*\*\* Settings \*\*\*

Library ../pycode/Common\_Exec.py

Resource 关键字/关键字index.txt

Resource 配置信息/configIndex.txt

Library ../pycode/Common\_Excel.py

\*\*\* Test Cases \*\*\*

test\_showCityName

[Documentation] /areaplug/showCityName

# 测试场景

${testScene} Set Variable showCityName

# 请求方法

${method} getTestHttpMethod ${testDataFile} ${testScene}

执行测试 ${testScene} ${method}

test\_getScheduleFlags

[Documentation] /ManageSchedule/getScheduleFlags

# 测试场景

${testScene} Set Variable getScheduleFlags

# 请求方法

${method} getTestHttpMethod ${testDataFile} ${testScene}

执行测试 ${testScene} ${method}

test\_calendar

# 测试场景

${testScene} Set Variable calendar

# 请求方法

${method} getTestHttpMethod ${testDataFile} ${testScene}

执行测试 ${testScene} ${method}

http\_Request.txt  
在各测试场景中会根据excel中的测试用例记录数目去批量调用http\_Request下的sendHttpRequest执行http接口测试。在sendHttpRequest中会根据caseNo去excel中查询相应测试数据，并发送对应的http请求到相应http接口中。收到响应后，与excel中的预期响应码和响应内容做比对。

\*\*\* Settings \*\*\*

Library ../pycode/Common\_Exec.py

Library ../pycode/Common\_Excel.py

Resource 关键字/关键字index.txt

\*\*\* Test Cases \*\*\*

sendHttpRequest

# 获取测试用例数据

${method} ${url} ${caseNo} ${testName} ${dict\_params} ${expectCode} ${expectCotent}

... getTestData ${testDataFile} ${testScene} ${Host} ${caseNo}

# 设置用例说明

Set Test Documentation ${testName}

# 请求头

${headers} 获取请求头

#根据method发送对应的http请求

${actualCode} ${actualContent} send\_HttpRequest ${url} ${dict\_params} ${headers} ${method}

# 响应码比对

Should Be Equal ${actualCode} ${expectCode}

# 响应内容比对

Should Be Equal ${actualContent} ${expectCotent}

关键字  
关键字模块主要是对一些复用逻辑的封装。

\*\*\* Settings \*\*\*

Resource ../配置信息/configIndex.txt

Library ../../pycode/Common\_Excel.py

Library ../../pycode/Common\_Exec.py

\*\*\* Keywords \*\*\*

获取请求头

${dict\_headers} Create Dictionary Host=${Host} User-Agent=${User-Agent} Accept=${Accept} Accept-Language=${Accept-Language} Accept-Encoding=${Accept-Encoding}

... Cookie=${Cookie} Connection=${Connection} Cache-Control=${Cache-Control}

Return From Keyword ${dict\_headers}

执行测试

[Arguments] ${testScene} ${method} # 测试场景|请求方法

# 获取用例数目

${case\_num} getTestCaseNum ${testDataFile} ${testScene}

# 获取当前时间

${execTime} getCurtime

#批量执行testScene测试场景下的用例

${status} batch\_Call ${httpTestSuite} ${httpRequestTestCase} ${testScene} ${case\_num} ${testCaseReportPath}

... ${execTime}

log ${status}

# 合并报告

${status} mergeReport ${testScene} ${testCaseReportPath} ${execTime}

log ${status}

# 清理合并前的报告

cleanLogs ${testScene} ${testCaseReportPath}

配置信息  
配置信息中存储配置信息以及通讯头的信息。通讯头中有cookie，保存有登录信息，通讯头的部分涉及隐私，故这部分数据不放出来了。  
config.txt

\*\*\* Settings \*\*\*

\*\*\* Variables \*\*\*

${testDataFile} E:/llf\_58TestSuites/jz\_webIntergration/robot\_code/testData/testData.xlsx # 测试数据

${httpRequestTestCase} sendHttpRequest # http请求用例模板

${httpTestSuite} E:/llf\_58TestSuites/jz\_webIntergration/robot\_code/rfcode/http\_Request.txt # http请求测试套件

${testCaseReportPath} E:/llf\_58TestSuites/jz\_webIntergration/robot\_code/report/TestCaseReport # 业务用例测试报告路径

RequestHeaders.txt

\*\*\* Settings \*\*\*

Documentation 请求头信息

\*\*\* Variables \*\*\*

${Host} \*\*\*\*\*\*\* # 服务器主机

${User-Agent} Mozilla/5.0 (Windows NT 6.1; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0 # 浏览器代理

${Accept} text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

${Accept-Language} en-US,en;q=0.5

${Accept-Encoding} gzip, deflate

${Cookie} \*\*\*\*\*\*\*\*\*\*\*\*

${Connection} keep-alive

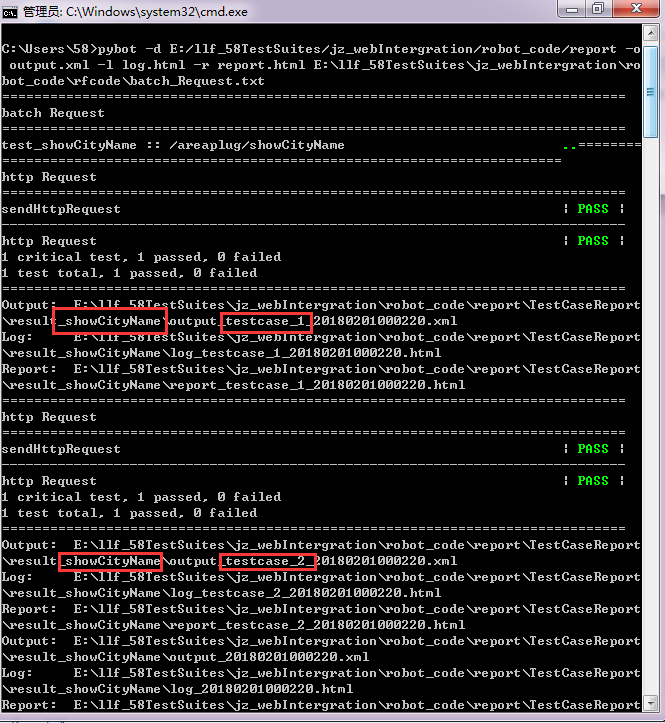
${Cache-Control} max-age=0

${Upgrade-Insecure-Requests} \*\*\*

* testData  
  该目录下存放测试数据excel文件。

执行测试

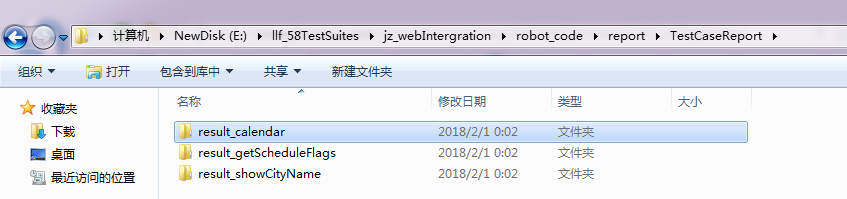
pybot -d E:/llf\_58TestSuites/jz\_webIntergration/robot\_code/report -o output.xml -l log.html -r report.html E:\llf\_58TestSuites\jz\_webIntergration\robot\_code\rfcode\batch\_Request.txt



执行测试

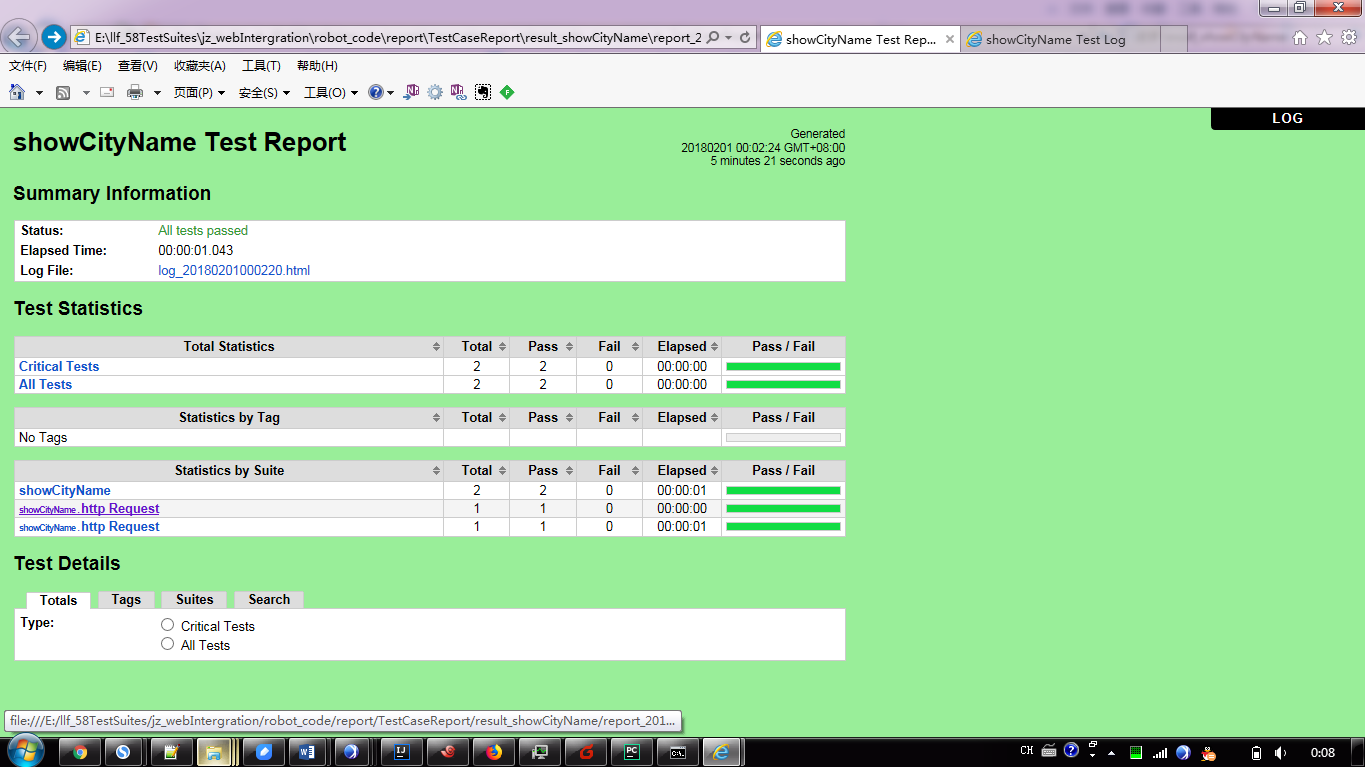
可见，showCityName测试场景已根据excel中的用例条数批量执行了测试。

进入TestCaseReport目录，可以看到已根据测试场景分别生成了对应目录的测试报告：

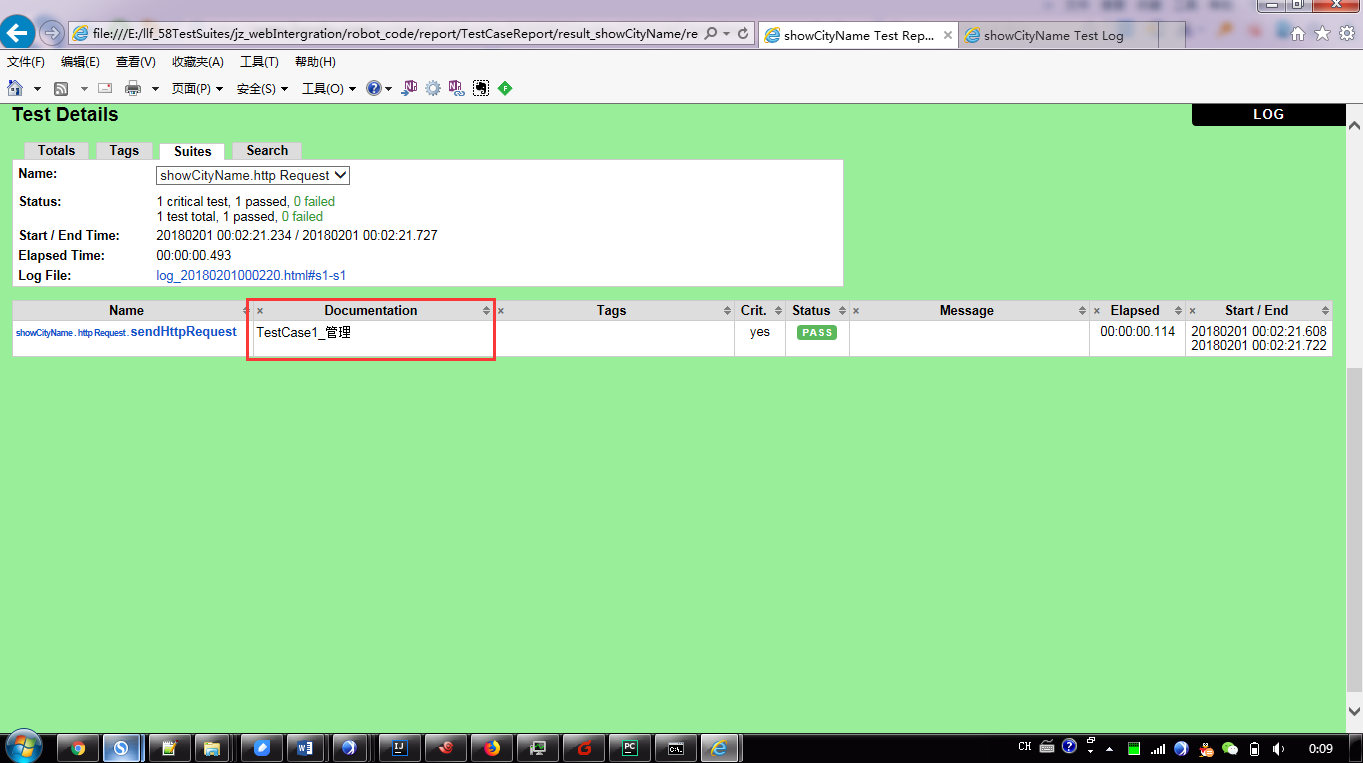


各测试场景的报告存在相应目录中

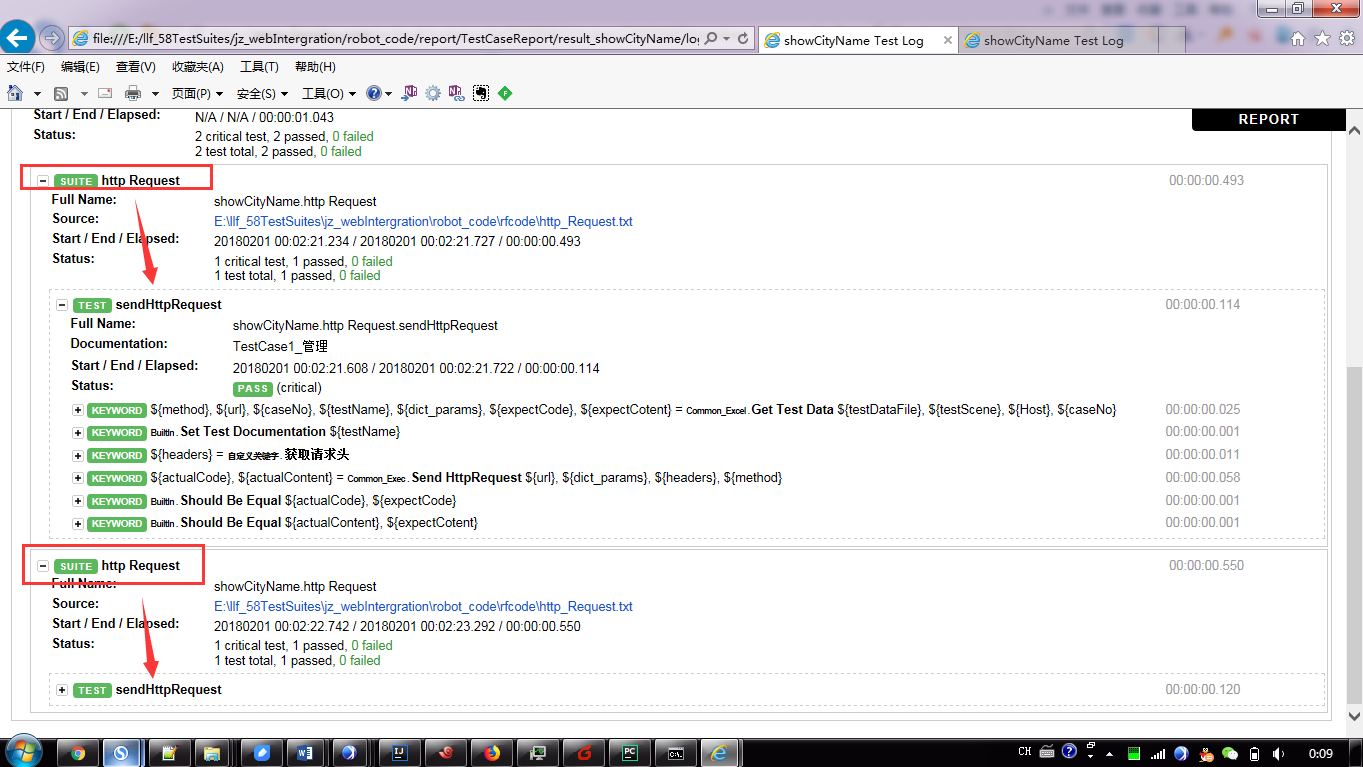
进入showCityName目录，打开最新生成的该场景测试报告：



showCityName场景测试报告



根据说明列辨别是哪条用例的报告数据



sendHttpRequest被批量调用