### 前言

&此文档作为目前主流的自动化框架参考,只写了本地实现，没有写持续集成持续集成可参考（主流自动化框架持续集成）这份文档

&涉及框架虽多，但是都是基于excel管理测试用例，目前我看了很多博客和书籍也是用excel比较多，毕竟没到测试开发的阶段，否则就可写前后端实现测试平台

&此文档讲到多个框架的配置，必定会有冗余的部分，比如用到的框架可能是java或python/每个框架用excel，那么excel定义的东西都一样的，不一样的无非是语法

### 修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修订人** | **版本说明** | **版本修订日期** |
| 1.0 | 朱财翔 | 添加robotframework  添加pytest  添加jmeter+ant | 2018.6.7 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 1.涉及持续集成框架

#### 1.1基于Python

Robotframework --CX

Pytest---CX

#### 1.2基于Java

Testng

Jmeter+ant--CX

### 2.robotframework实现 ui自动化

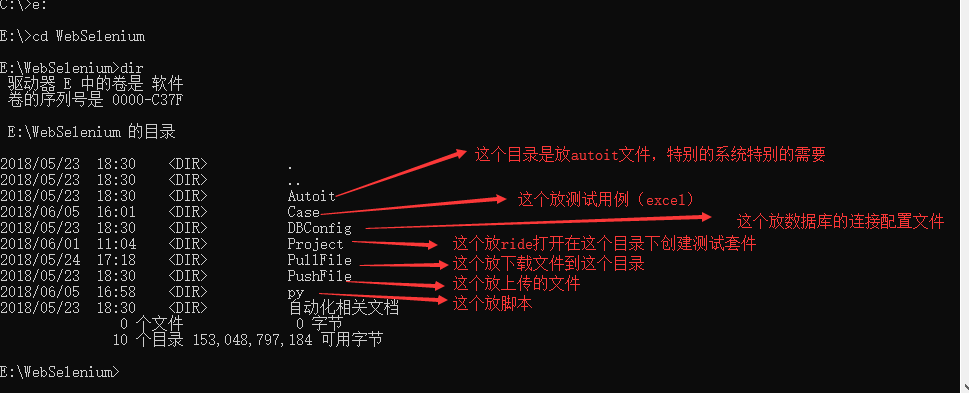
#### 1.1安装教程

& <https://blog.csdn.net/ywyxb/article/details/64126927>

可参考这个地址安装，安装不多讲，网上资料一大推。

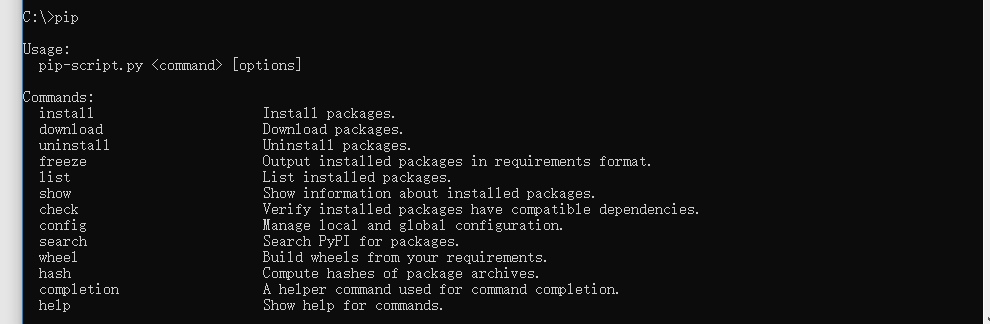
#### 1.2创建一个工程

&工程下有哪些目录，这样放文件比较明确，如下



&创建好，这时候就得写脚本主题是UI自动化，所以用selenium库，还有其他得辅助库

，cd c:\Python27\Scripts\,执行easy\_install pip, 安装好pip后dos执行pip，成功如下，不成功百度找原因



&这时候装下UI自动化常用的库

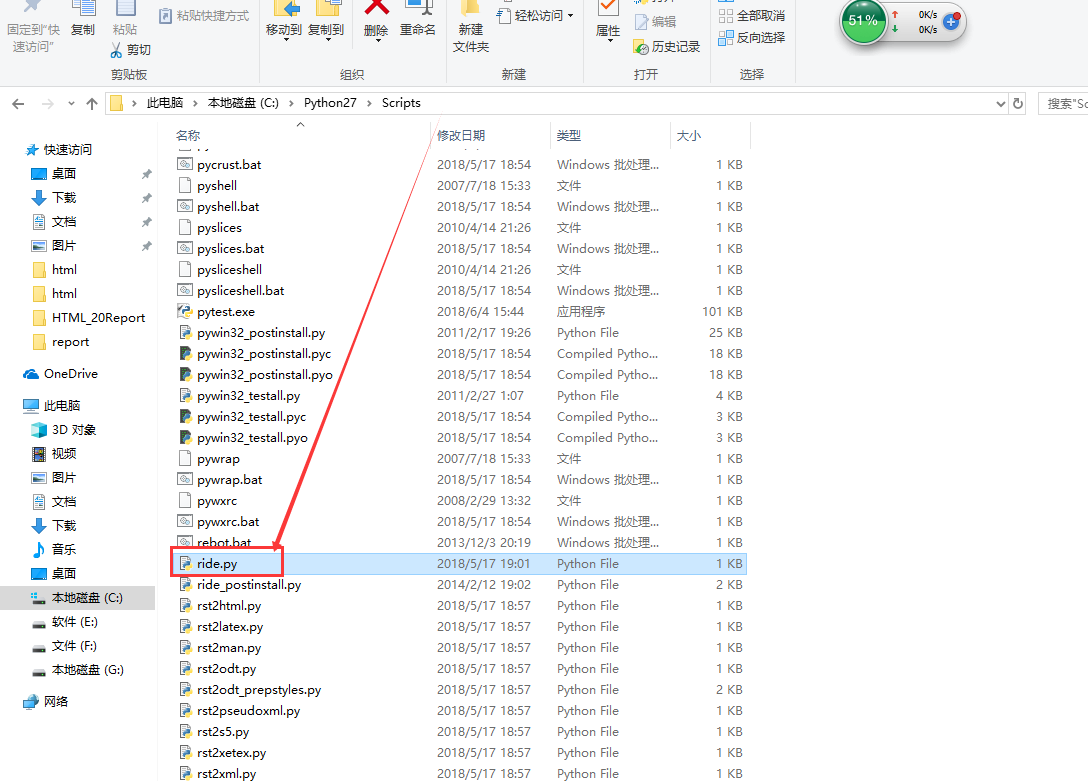
Pip install selenium

Pip install pymysql

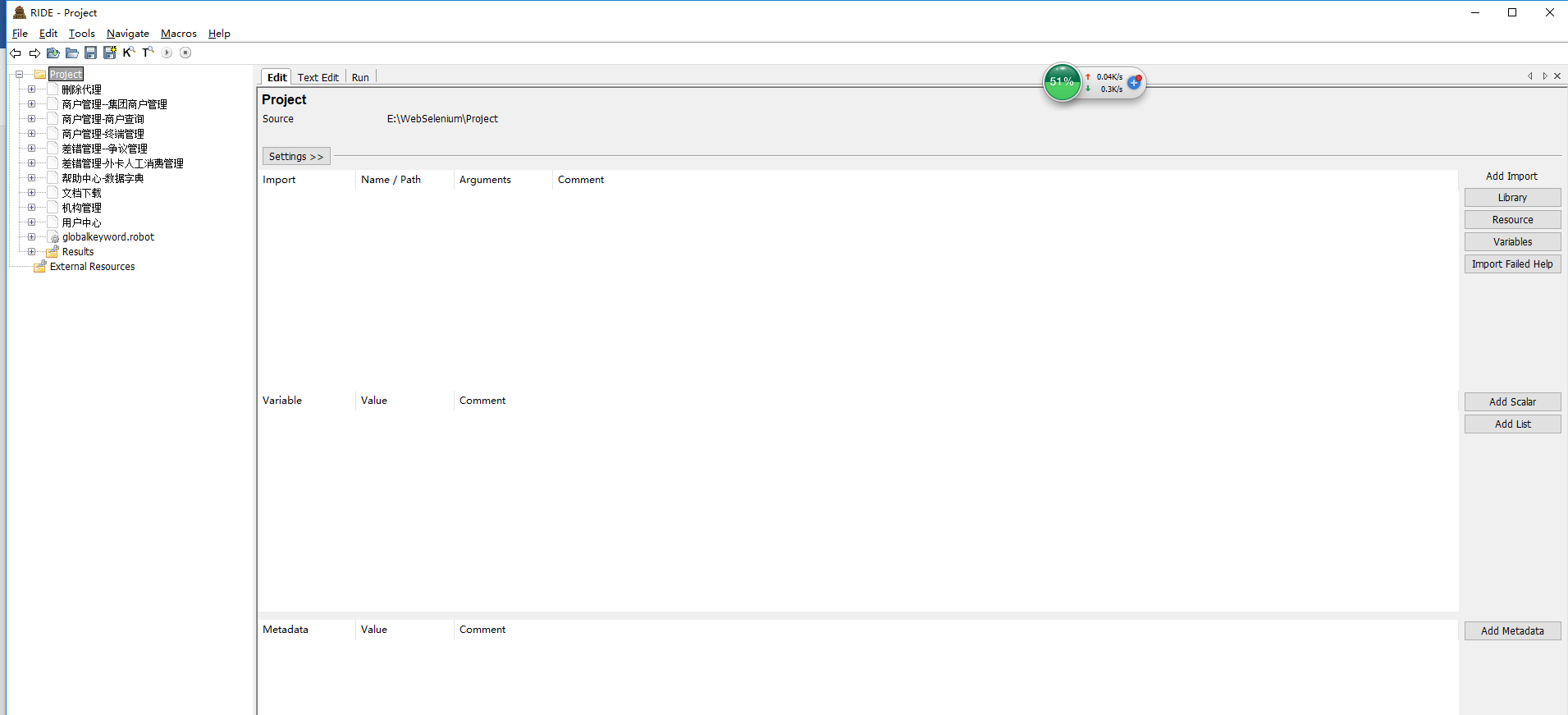
Pip install cx\_Oracle

#### 1.3打开ride

&到你本地对应的地方打开这个ride.py

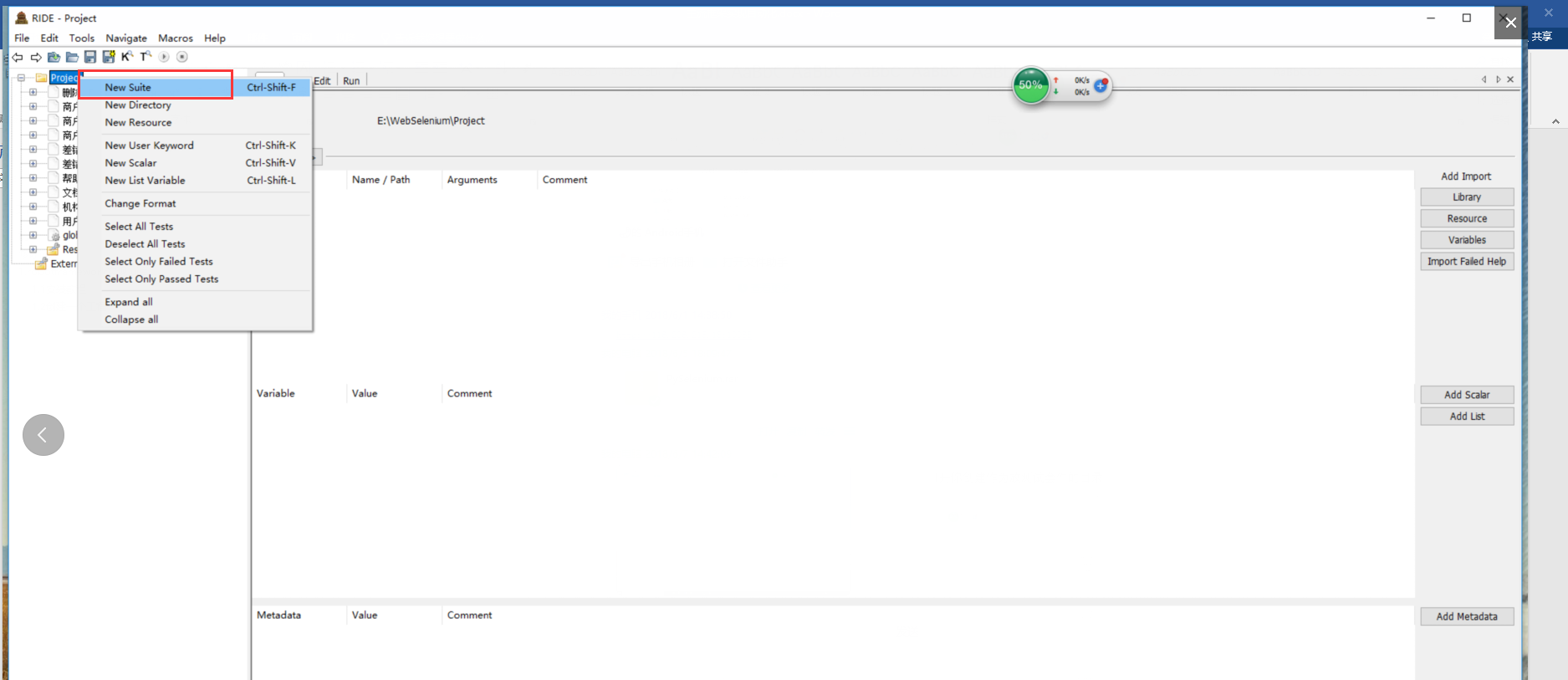


&打开后的样子，我的是已有文件了，如果你没有的话，打开你创建作为放测试套件的目录

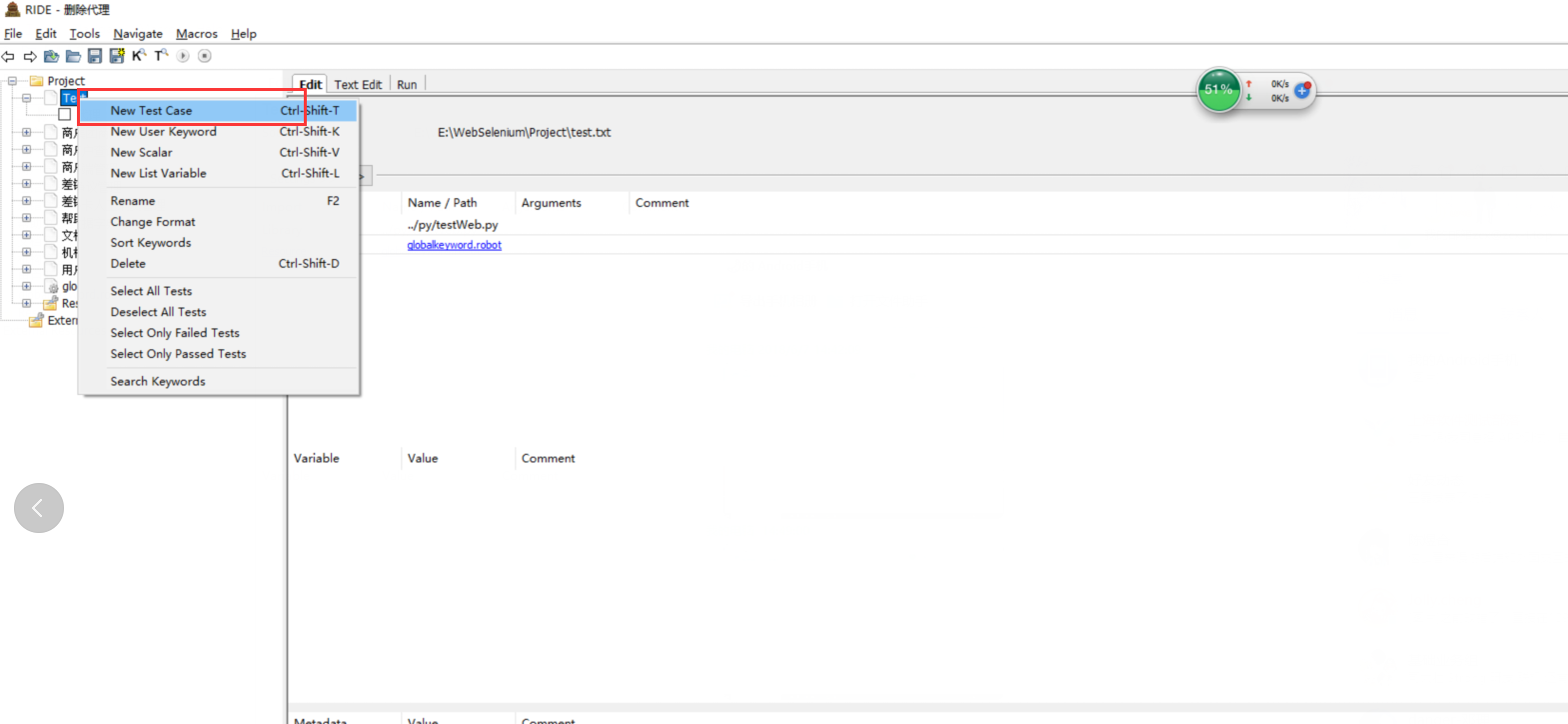


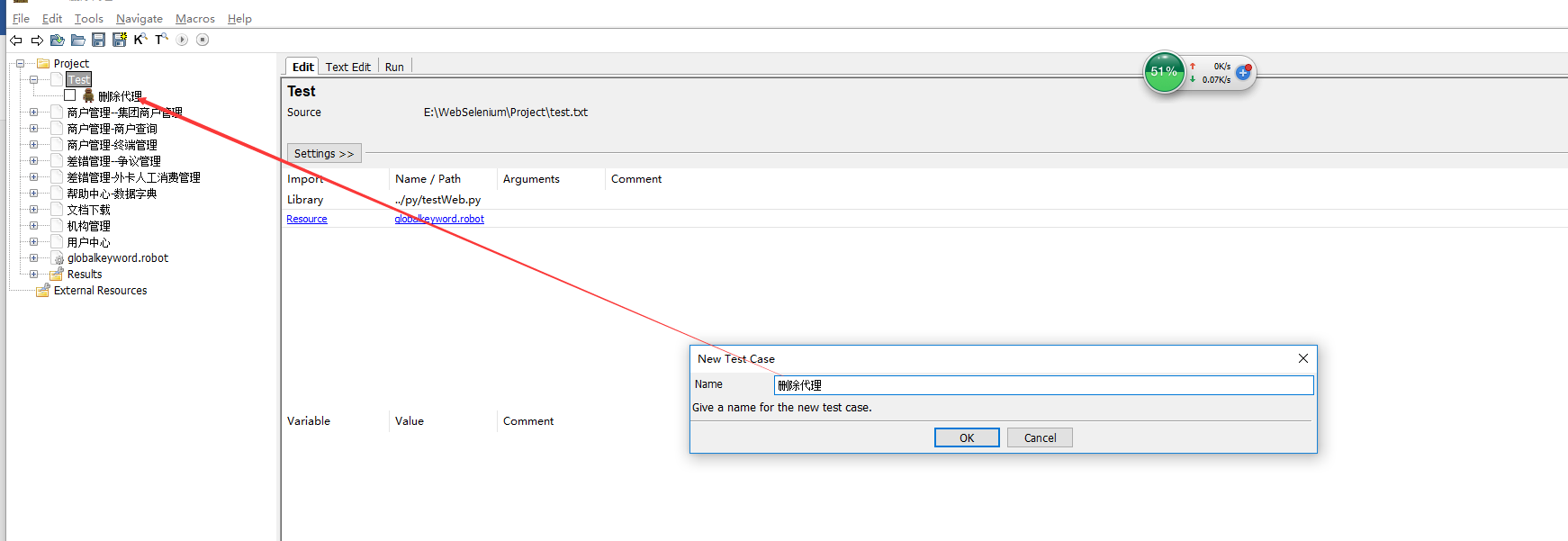
1.3新建一个测试用例

&新建一个Test测试套件

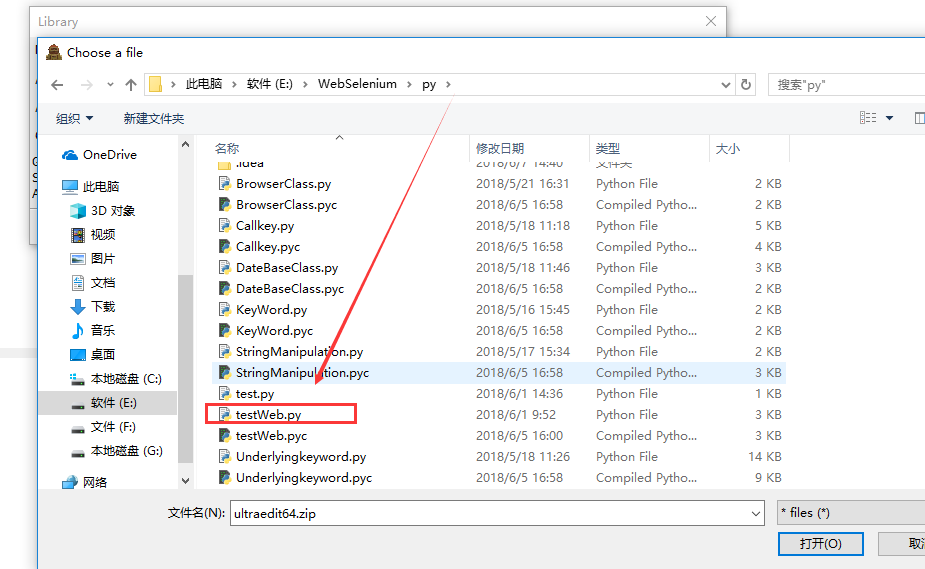


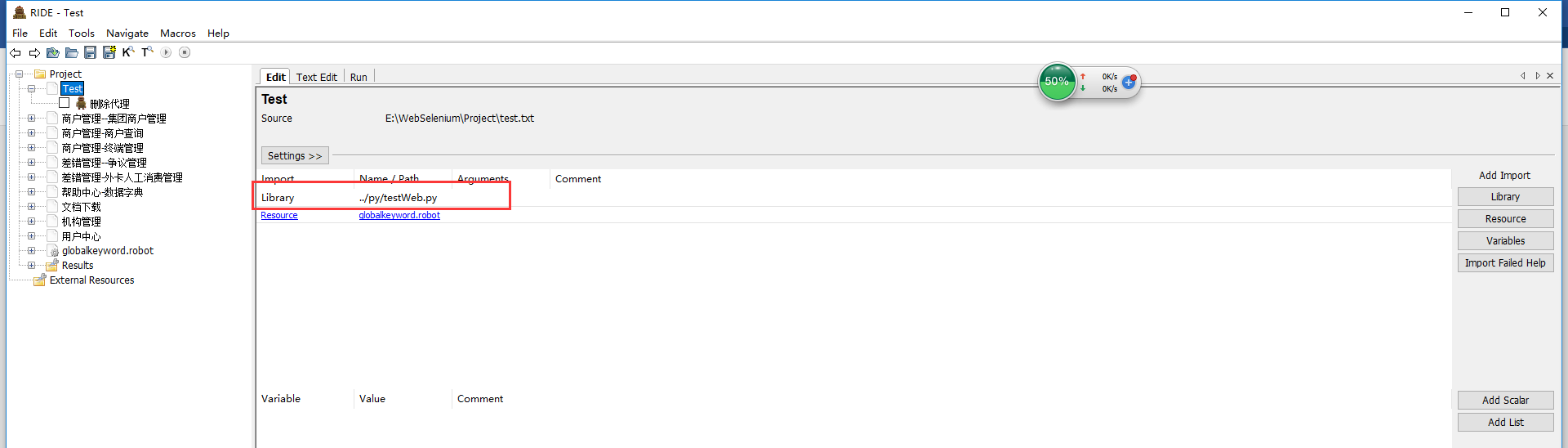
&在测试套件上再创建一个测试用例





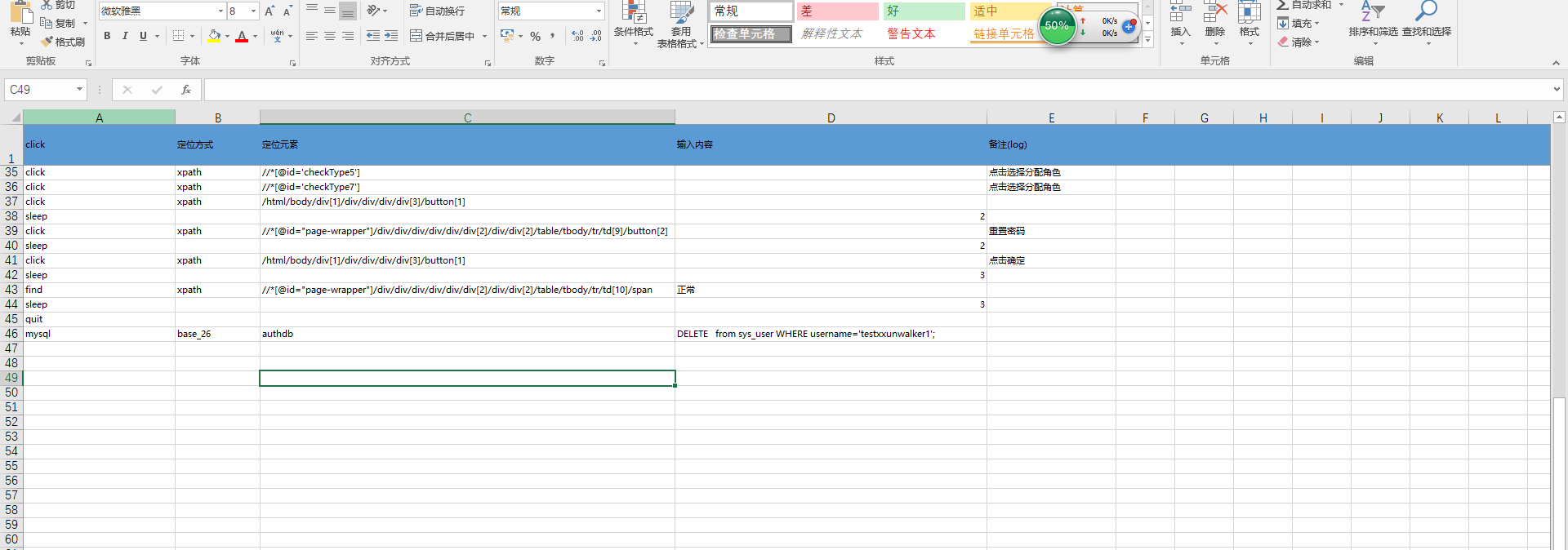
#### 1.4导入脚本





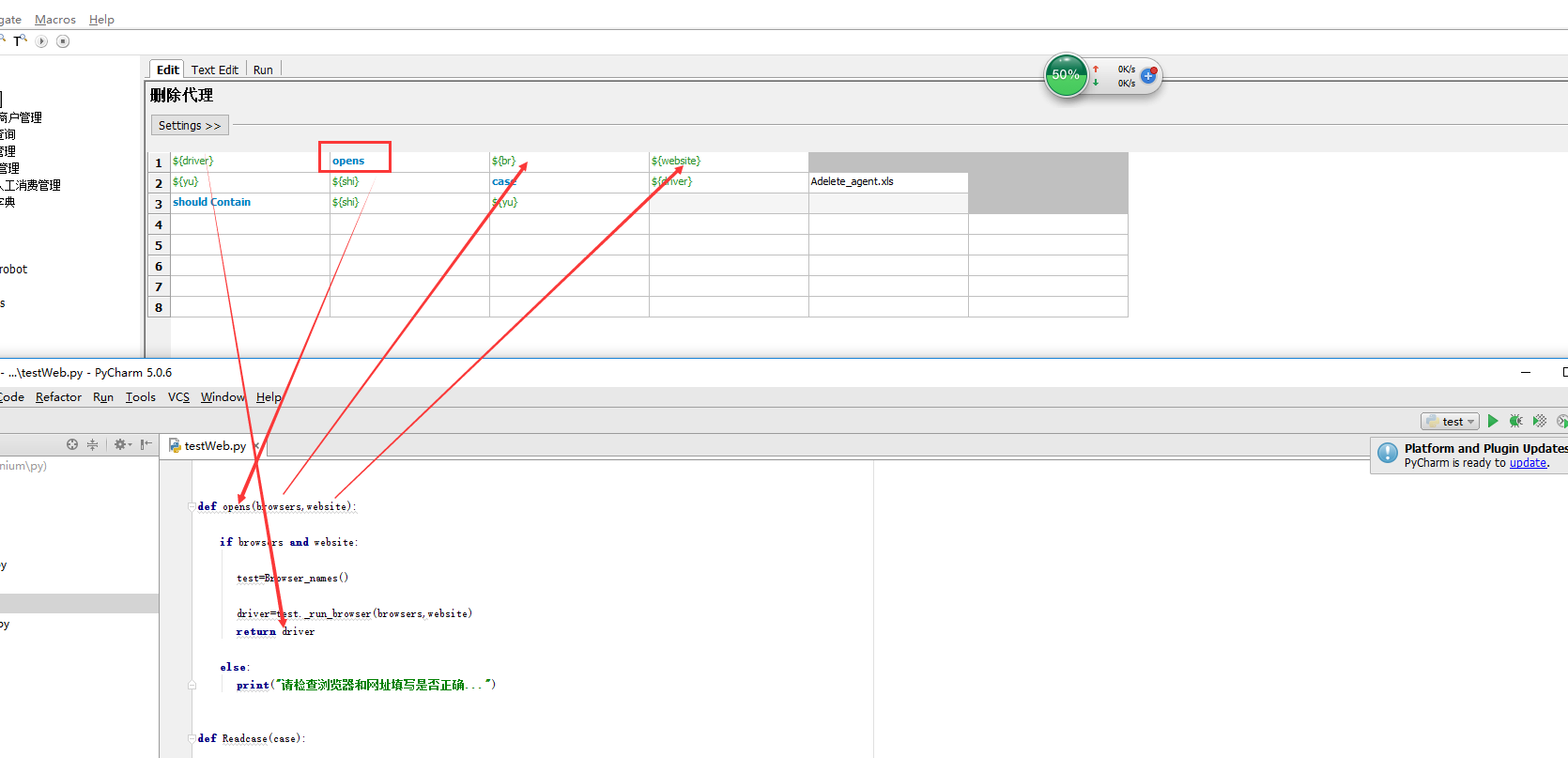
#### 1.5脚本实现思路

&定义一个自动化用例模板，封装关键字，参数化实现，所以导入主方法testweb.py

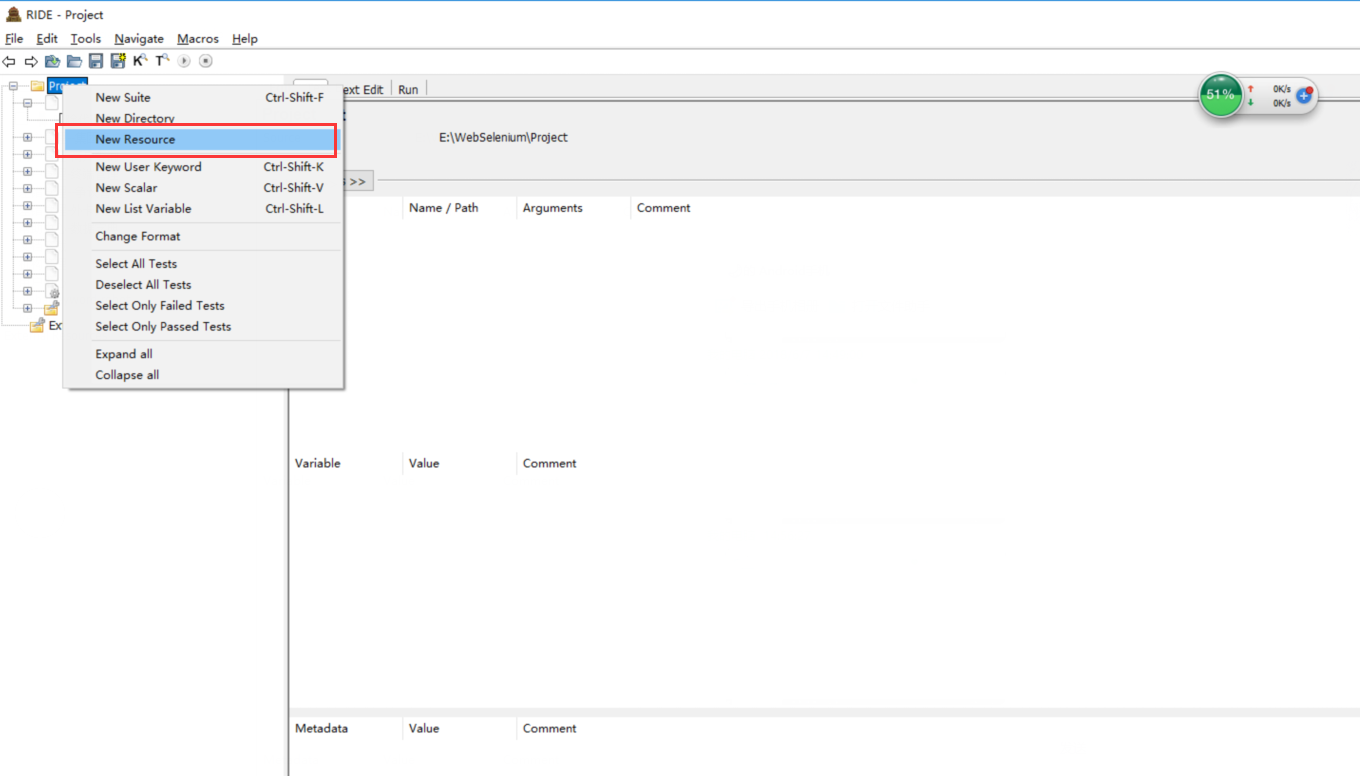


#### 1.6全局变量和用例填写

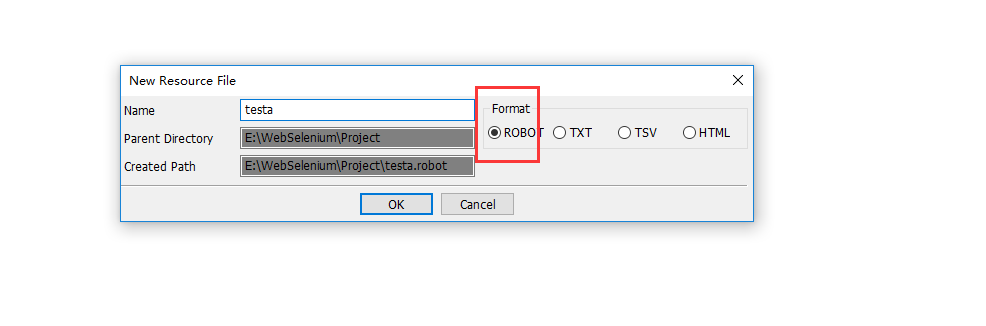
&ride中case填写规则，执行方法索引到指定excel



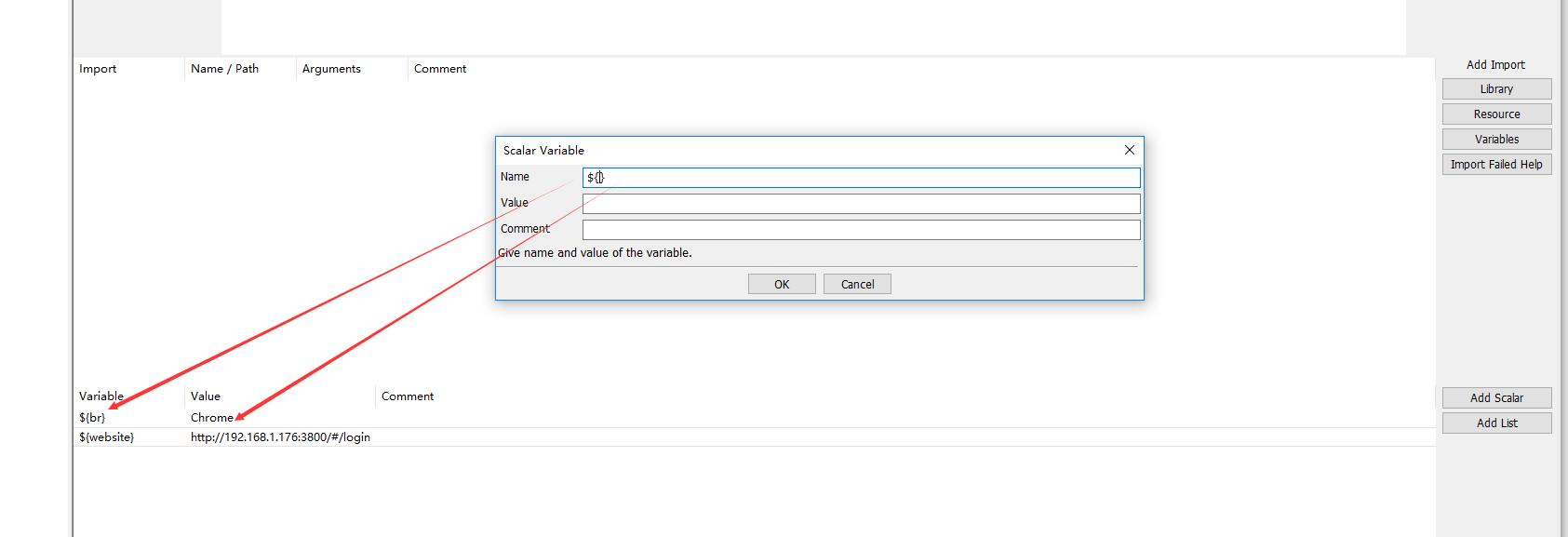
添加一个New Resource--起一个名字—选择robot格式文件



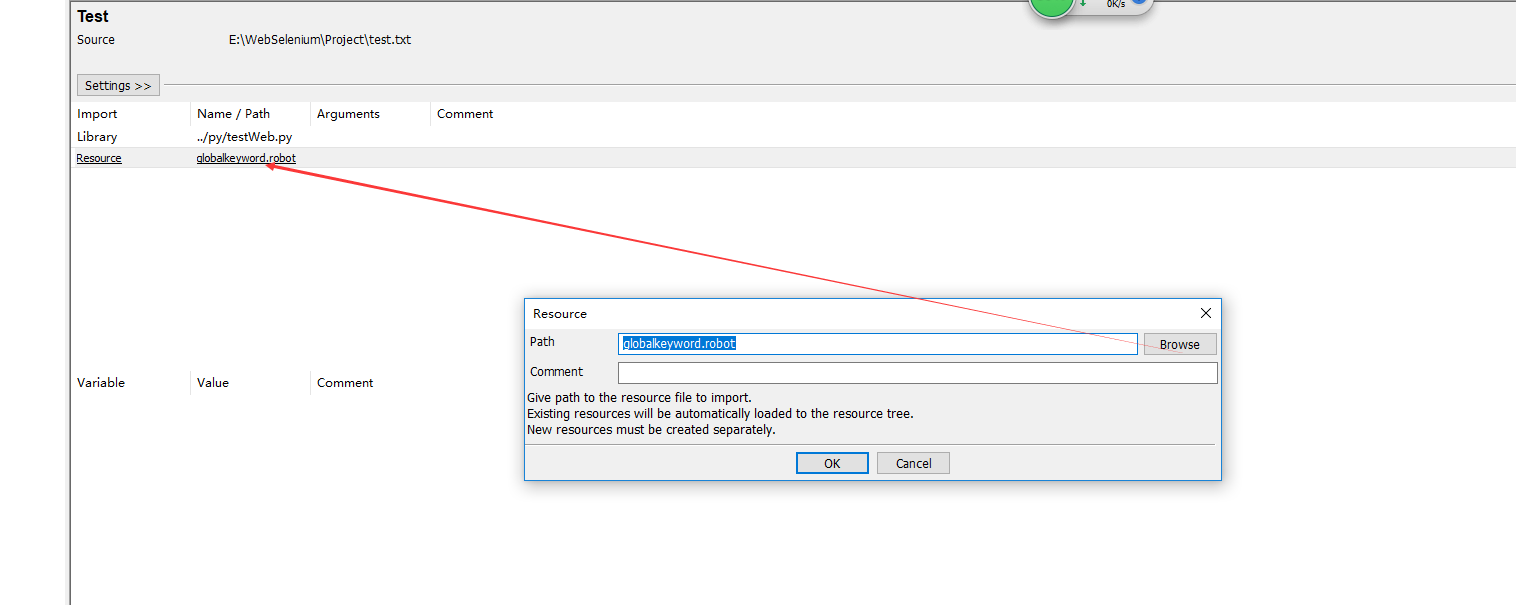
确认~



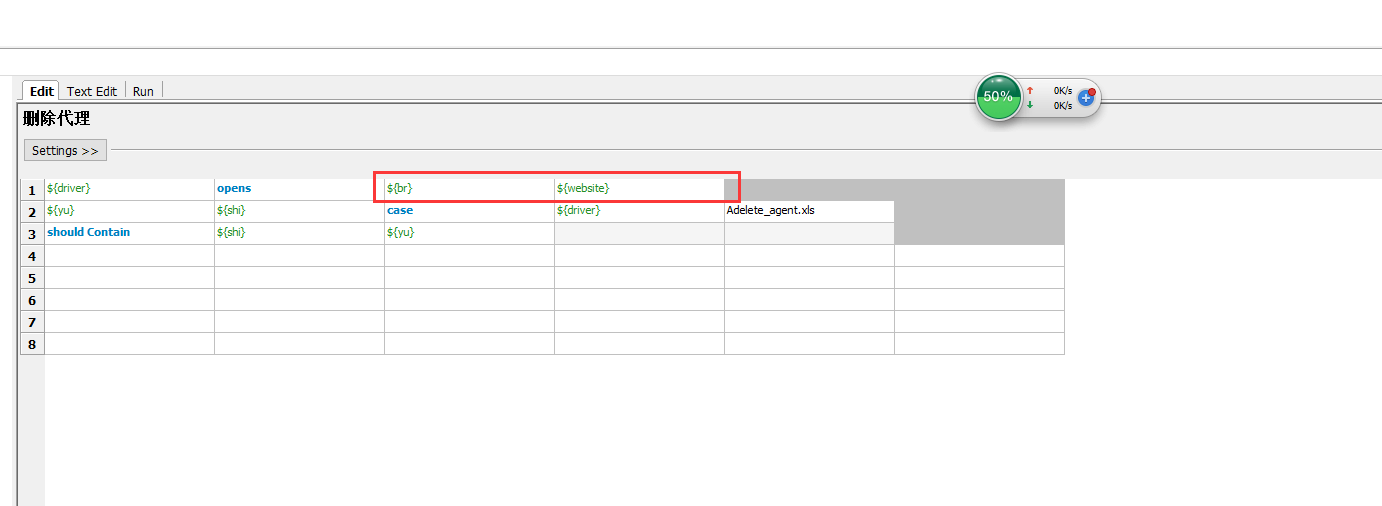
&配置下变量和值



&导入全局变量文件—选择创建的文件

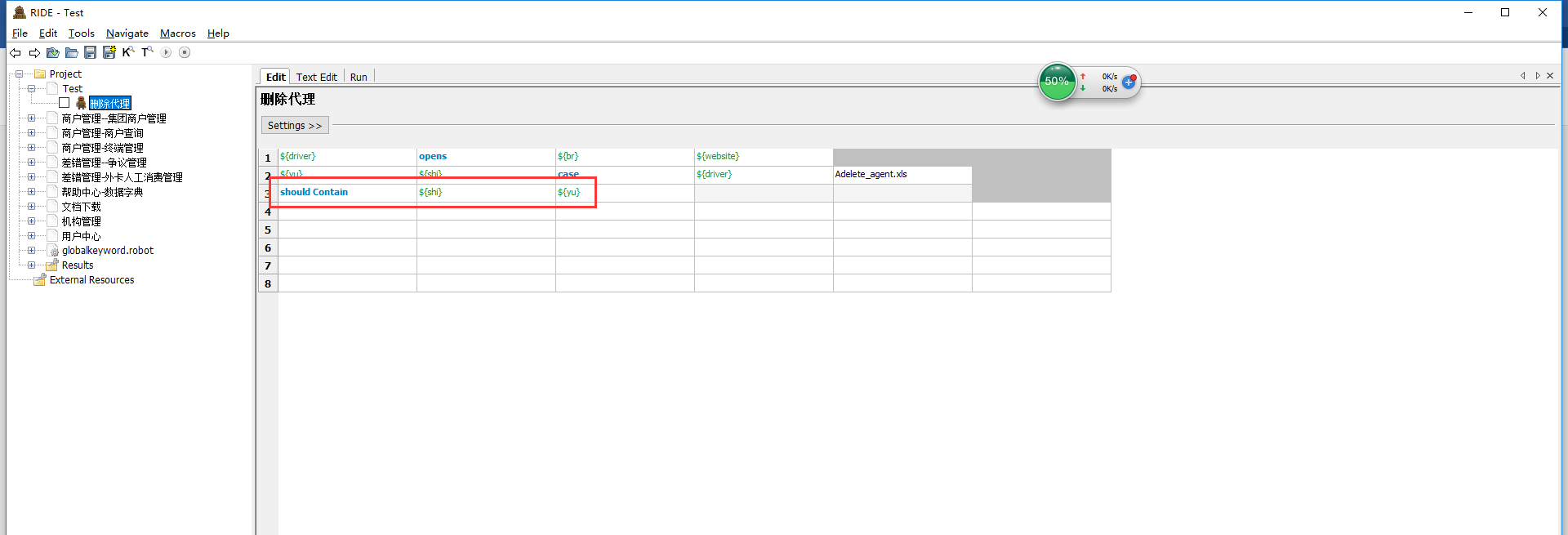


&使用创建好的变量



#### 1.7断言

&通过case方法return出实际和预期结果，然后ride自带的断言should Contain判断下这两个值是否有包含的关系得到pass/fail



#### 1.8多线程运行用例

&执行以下命令，以安装pabot(注意版本以下python为例)：

Pip install  robotremoteserver==1.0.1

pip install robotframework-pabot==0.22

&安装后，执行(10代表线程数)

pabot --processes 10 --outputdir results project路径/

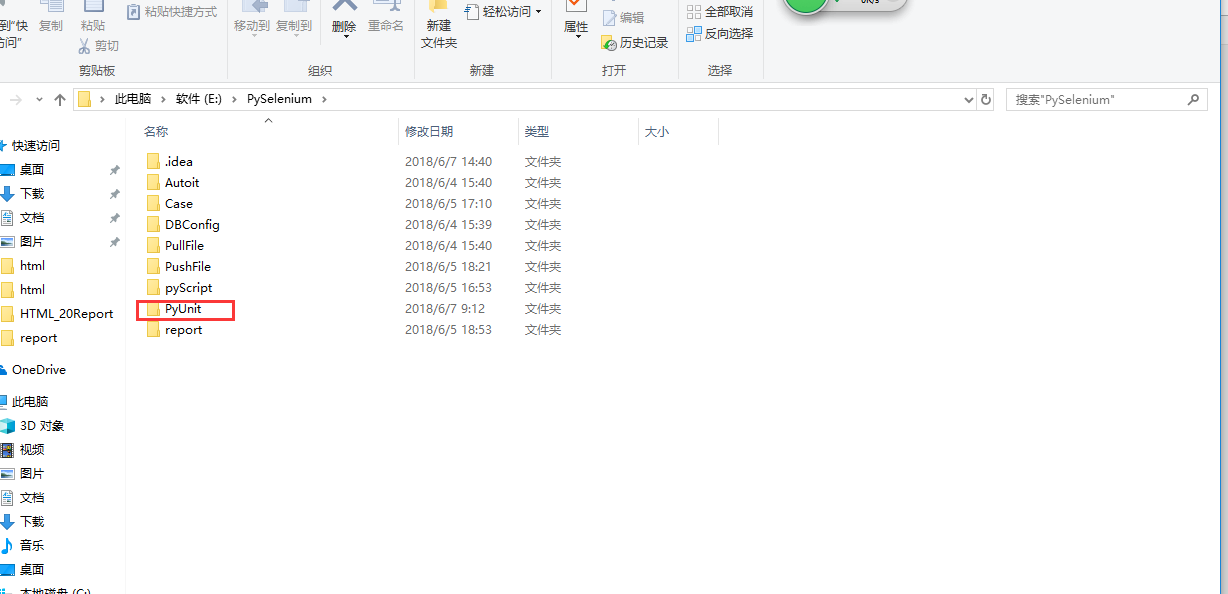
### 3.pytest实现UI自动化

#### 1.1为啥不用unittest

Pytest相对Unittest更加灵活一点，unittest需要写类继承unittest.TestCase类,pytest不需要可写可不写类，也支持生成html测试报告

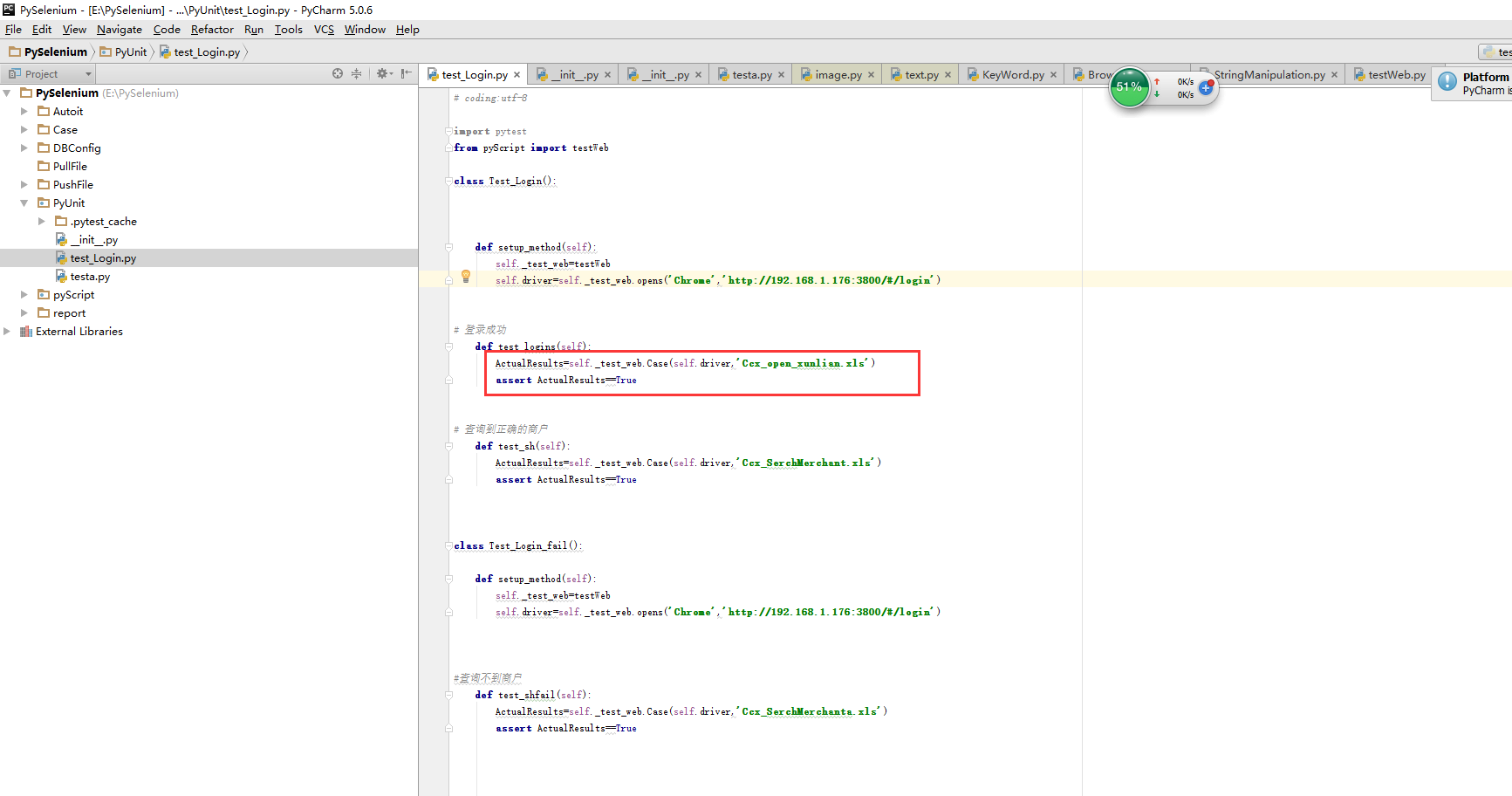
#### 1.2.创建工程

&对比上面robot的工程，增加一个PyUnit目录和report，这里主要放单元测试框架其他实现思路一致,report放执行后生成的html报告



3.新建测试用例

&可以看出单元测试框架命名规则风格其实都是一致的，调用“底层“写的一个方法去读execel运行测试用例



#### 1.3pytest运行用例规则

&深入研究请百度，下面只让你能独立写一个自动化工程的目的

1.文件已 test\_开头 test\_\*.py

2.类已Test\_开头 class Test\_login()

3.方法已test\_开头 def test\_login()

#### 1.4初始化和清理的方法使用

百度还是挺细节化的<https://www.cnblogs.com/ranxf/p/7766229.html>

* fixture函数可以使用在测试函数中，测试类中，测试文件中以及整个测试工程中。
* fixture支持模块化，fixture可以相互嵌套
* fixture支持参数化
* fixture支持unittest类型的setup和teardown

1）模块级（setup\_module/teardown\_module）开始于模块始末：**在所有测试用例开始执行setup\_module和所有用例执行结束后执行teardown\_module**

2）类级（setup\_class/teardown\_class）开始于类的始末：**在当前测试类的开始与结束执行**

3）类里面的（setup/teardown）（运行在调用函数的前后）

4）功能级（setup\_function/teardown\_function）开始于功能函数始末（不在类中）：**用于每个测试用例开始执行时执行setup\_function和在每个用例执行结束后执行teardown\_function**

5）方法级（setup\_method/teardown\_method）开始于方法始末（在类中）：**在每个测试方法开始与结束执行**

&本次以UI自动化为例，只用到的setup\_method每次用例执行先调用浏览器打开网址

#### 1.5多种运行测试用例

&运行目录下所有测试用例

Dos下 到Pyunit下执行 py.test –html=需要生成的report路径

&运行执行某个py文件里的用例

Dos下 到Pyunit下执行 py.test 文件名.py –html=需要生成的report路径

&运行类里的某个方法

pytest -v test\_a.py::TestClass::test\_method

&运行指定的类中所有方法

pytest -v test\_a.py::TestClass

&组合运行

pytest -v test\_a.py::TestClass test\_b.py:: test\_method

#### 1.6断言

简单断言，用例执行完返回一个True或False assert==True,是True则通过，否则失败

### 4.Jmeter+ant实现接口自动化

#### 4.1安装jmeter 和ant

&环境变量配置百度去~

#### 4.2jmeter和ant关联

&将JMeter所在目录下extras子目录里的ant-JMeter-1.1.1.jar复制到ant的lib目录下

&修改Jmeter的bin目录下jmeter.properties文件的配置：

jmeter.save.saveservice.output\_format=xml

#### 4.3编写build文件

<?xml version="1.0" encoding="UTF-8"?>

<project name="ant-jmeter-test" default="run" basedir=".">

<tstamp>

<format property="time" pattern="yyyyMMddhhmm" />

</tstamp>

<!-- 需要改成自己本地的 Jmeter 目录-->

<property name="jmeter.home" value="E:\apache-jmeter-3.3" />

<!-- jmeter生成jtl格式的结果报告的路径-->

<property name="jmeter.result.jtl.dir" value="E:\apache-jmeter-3.3\report\jtl" />

<!-- jmeter生成html格式的结果报告的路径-->

<property name="jmeter.result.html.dir" value="E:\apache-jmeter-3.3\report\html" />

<!-- 生成的报告的前缀-->

<property name="ReportName" value="TestReport" />

<property name="jmeter.result.jtlName" value="${jmeter.result.jtl.dir}/${ReportName}${time}.jtl" />

<property name="jmeter.result.htmlName" value="${jmeter.result.html.dir}/${ReportName}${time}.html" />

<target name="run">

<antcall target="test" />

<antcall target="report" />

</target>

<target name="test">

<taskdef name="jmeter" classname="org.programmerplanet.ant.taskdefs.jmeter.JMeterTask" />

<jmeter jmeterhome="${jmeter.home}" resultlog="${jmeter.result.jtlName}">

<!-- 声明要运行的脚本。"\*.jmx"指包含此目录下的所有jmeter脚本-->

<testplans dir="E:\apache-jmeter-3.3\bin\testCase" includes="\*.jmx" />

<property name="jmeter.save.saveservice.output\_format" value="xml"/>

</jmeter>

</target>

<path id="xslt.classpath">

<fileset dir="${jmeter.home}/lib" includes="xalan\*.jar"/>

<fileset dir="${jmeter.home}/lib" includes="serializer\*.jar"/>

</path>

<target name="report">

<tstamp><format property="report.datestamp" pattern="yyyy/MM/dd HH:mm"/></tstamp>

<xslt

classpathref="xslt.classpath"

force="true"

in="${jmeter.result.jtlName}"

out="${jmeter.result.htmlName}"

style="${jmeter.home}/extras/jmeter-results-detail-report\_21.xsl" >

<param name="dateReport" expression="${report.datestamp}"/>

</xslt>

<!-- 因为上面生成报告的时候，不会将相关的图片也一起拷贝至目标目录，所以，需要手动拷贝 -->

<copy todir="${jmeter.result.html.dir}">

<fileset dir="${jmeter.home}/extras">

<include name="collapse.png" />

<include name="expand.png" />

</fileset>

</copy>

</target>

</project>

&配置相关的路径，然后dos下cd到有build的目录，执行ant即可

#### 4.4jmeter连接数据库包配置

&准备相关的包操作数据库JDBC

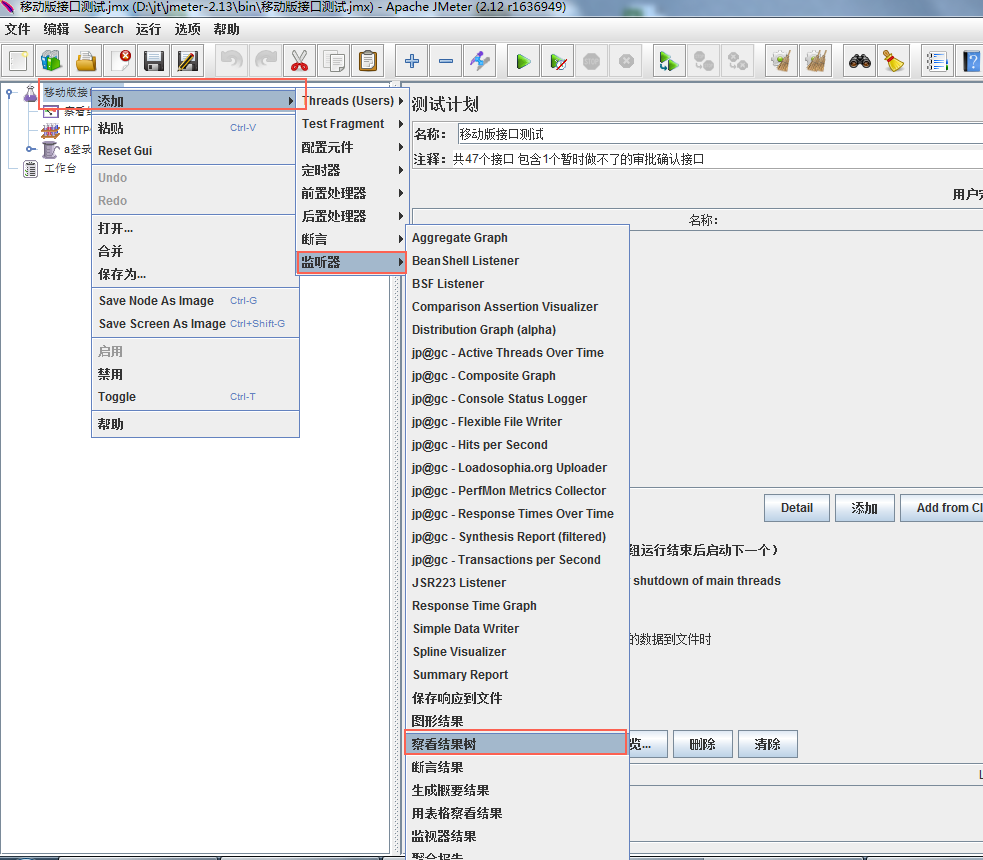
如果数据库是oracle，在客户端app/client/Administrator\product\12.1.0\client\_1\jdbc\lib下的ojdbc6.jar放在jmeter下的jmeter-2.12\lib目录下（注:我当前jmeter版本是2.12,不同版本和数据库用的包不一致，具体百度）

加载ojdbc6.Jar后才能使用

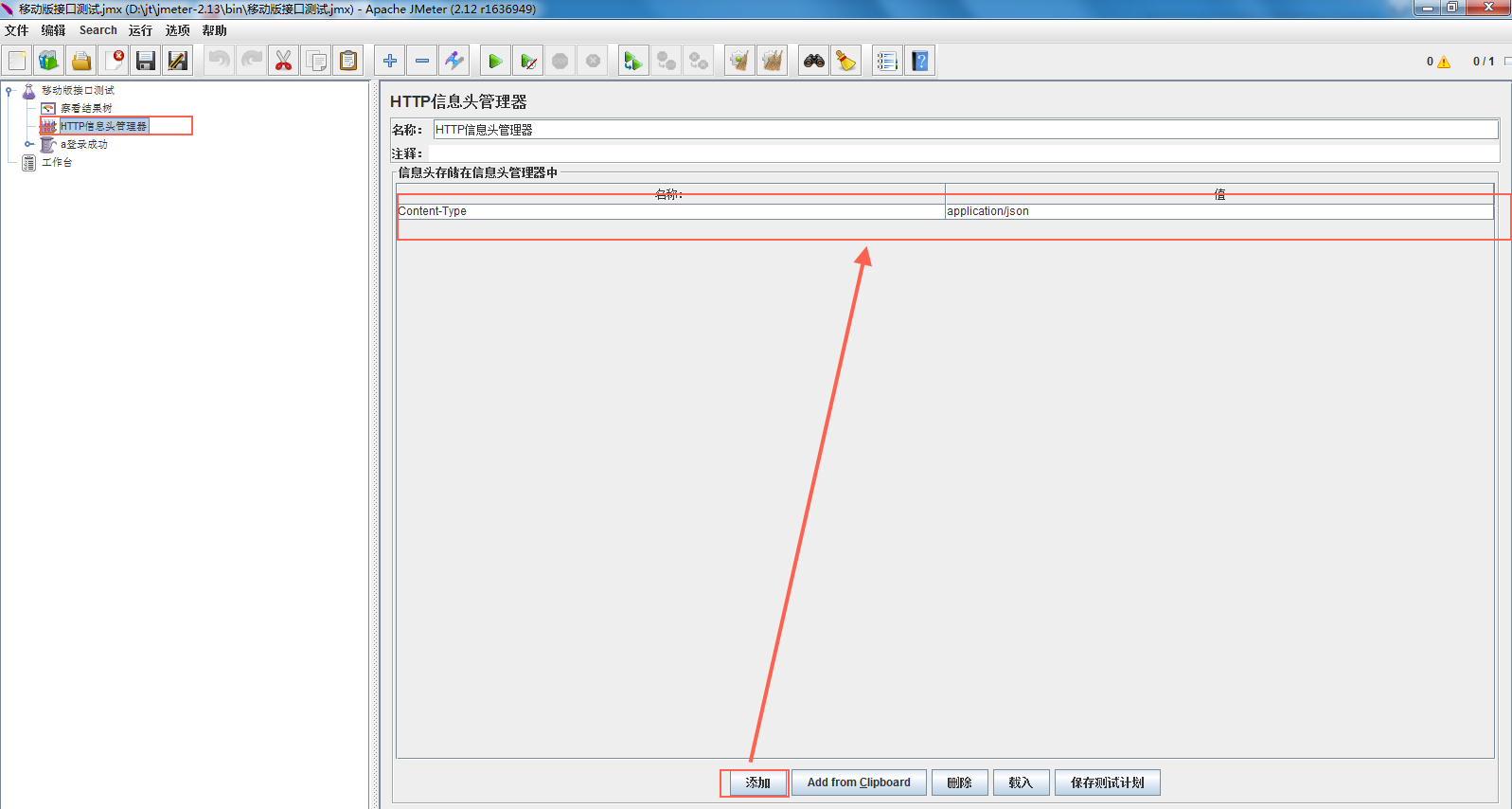
#### 4.5打开jmeter添加组件—发请求

4.5.1创建查看结果树、线程组、头文件、HTTP请求

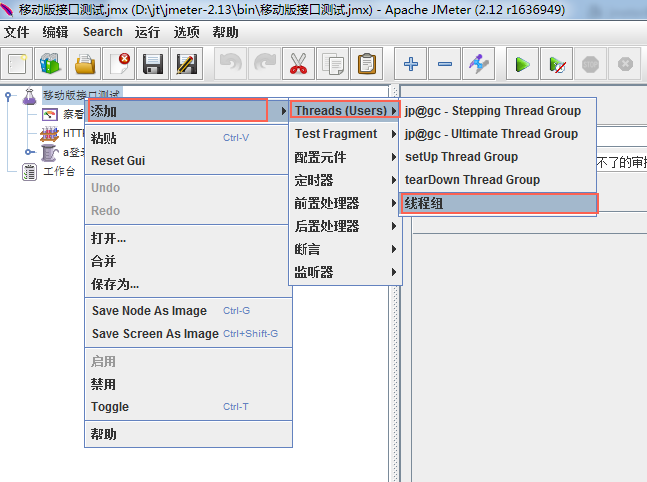
4.5.2 添加察看结果树：项【目名】--右击添加--【监听器】--【察看结果树】（如图）



4.5.2 添加HTTP信息头管理器: 【项目名】--右击添加--【配置元件】--【HTTP信息头管理器】注：HTTP信息管理器添加头文件 （如图）

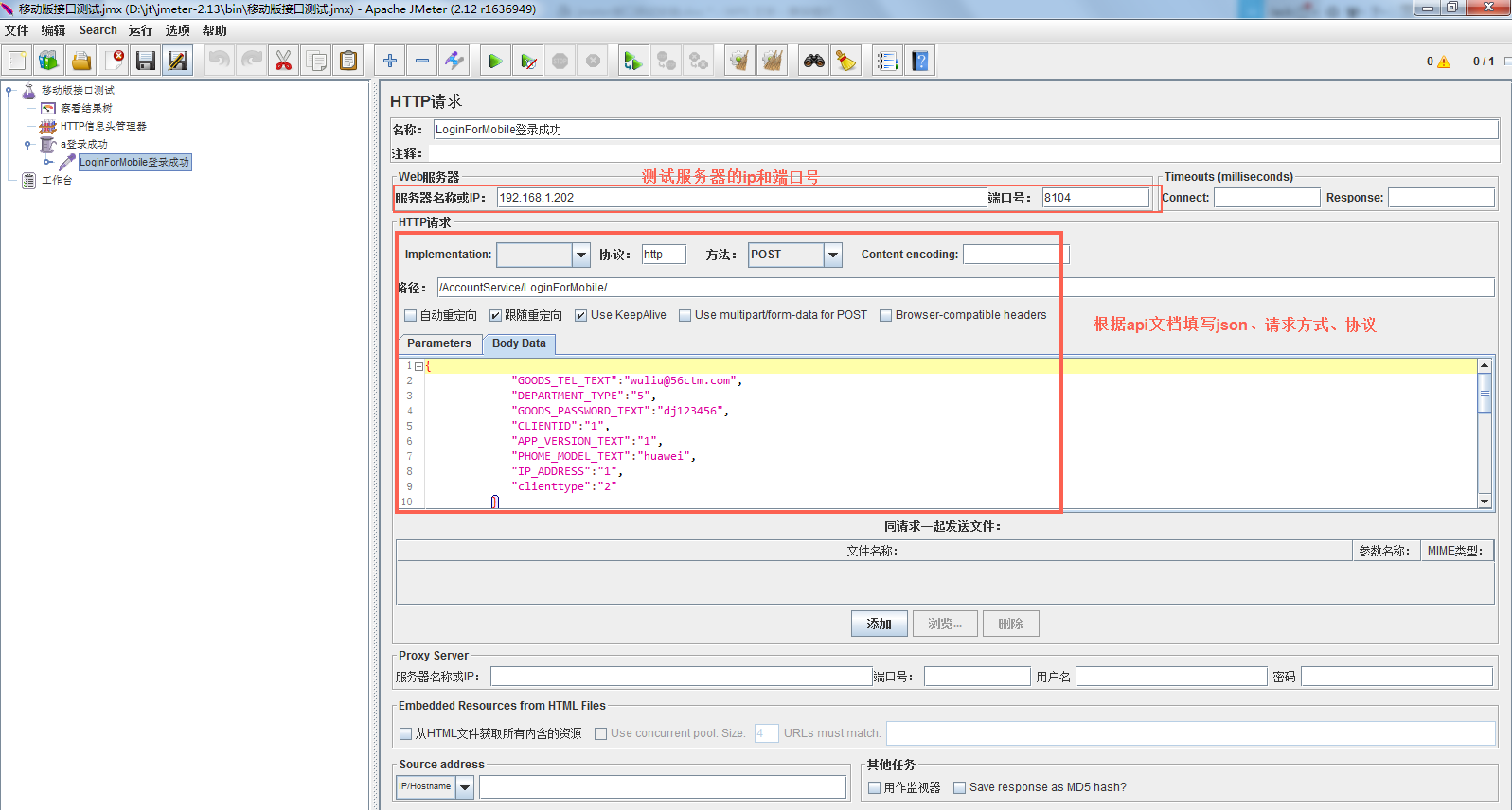


4.5.3 添加线程组：【项目名】--右击添加--【Threads(Users)】--【线程组】 （如图）

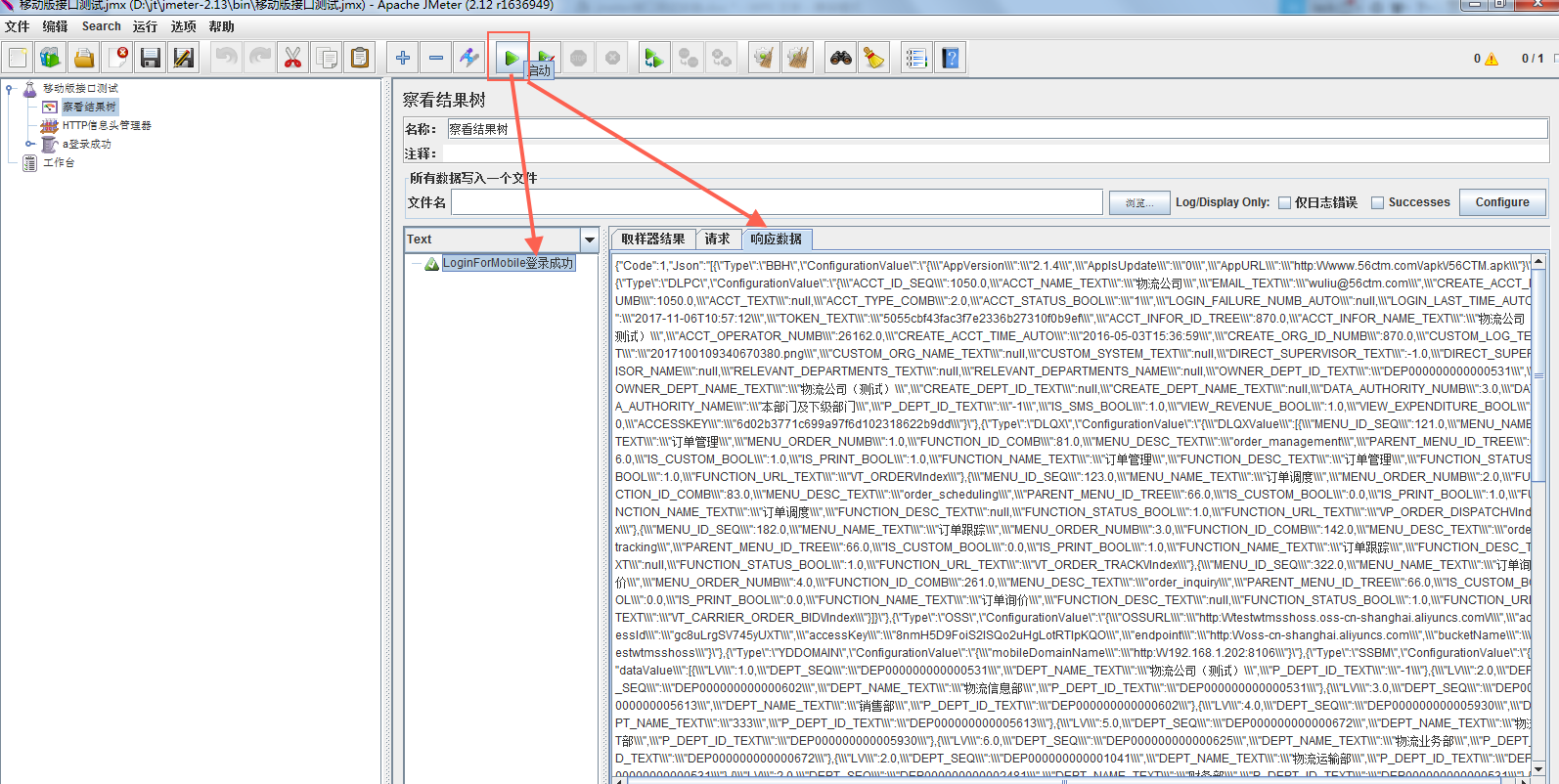


4.5.4 添加http请求，【线程组】--右击添加--【Sampler】--【HTTP请求】

#### 4.6 根据接口文档--登录请求（如图--登录接口）

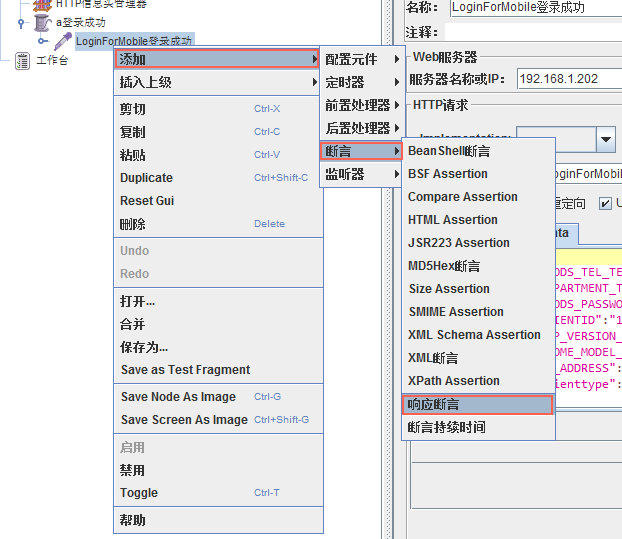


#### 4.7接口返回响应数据（如图）

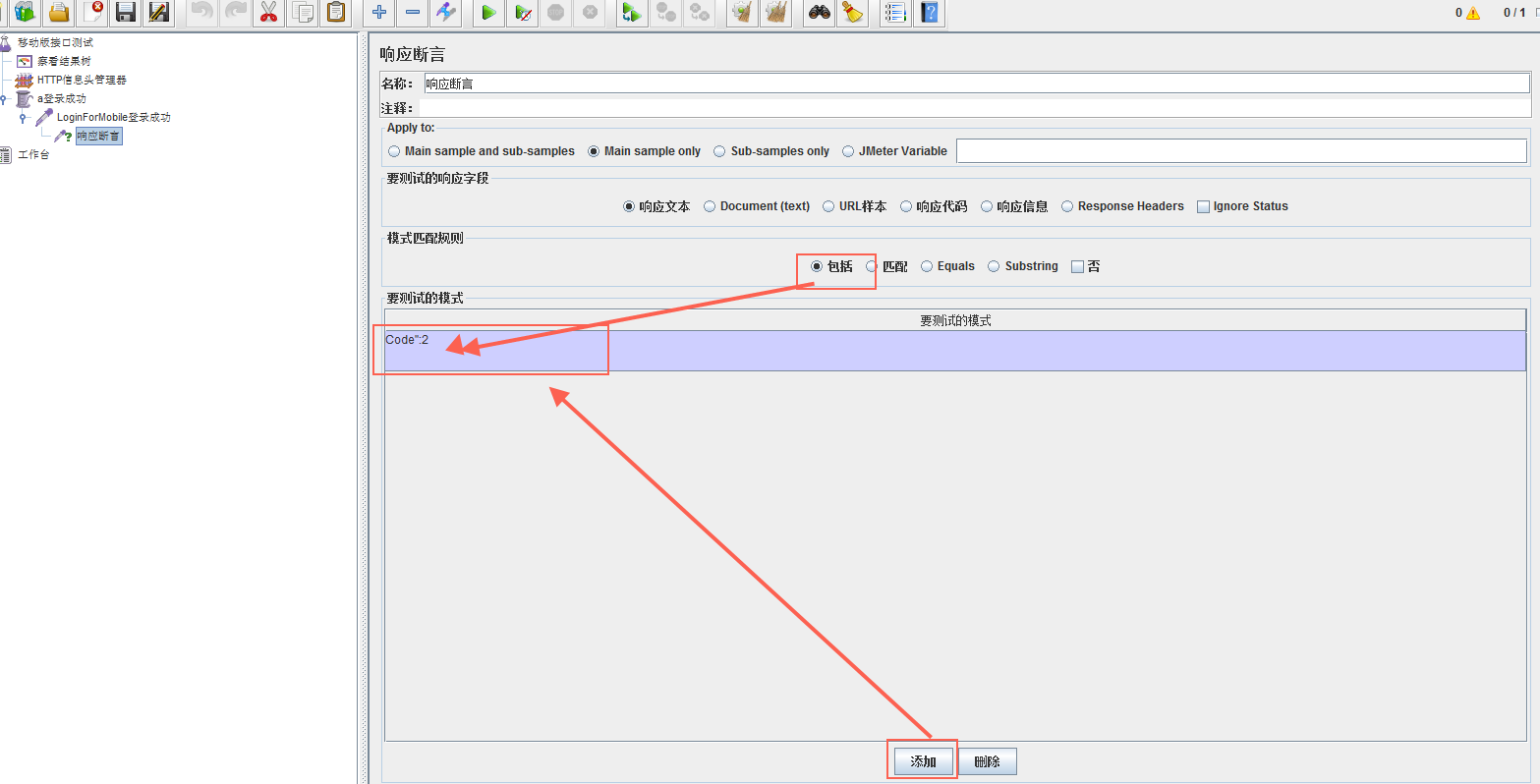


#### 4.8断言判断（接着登录接口）

**&**添加响应断言

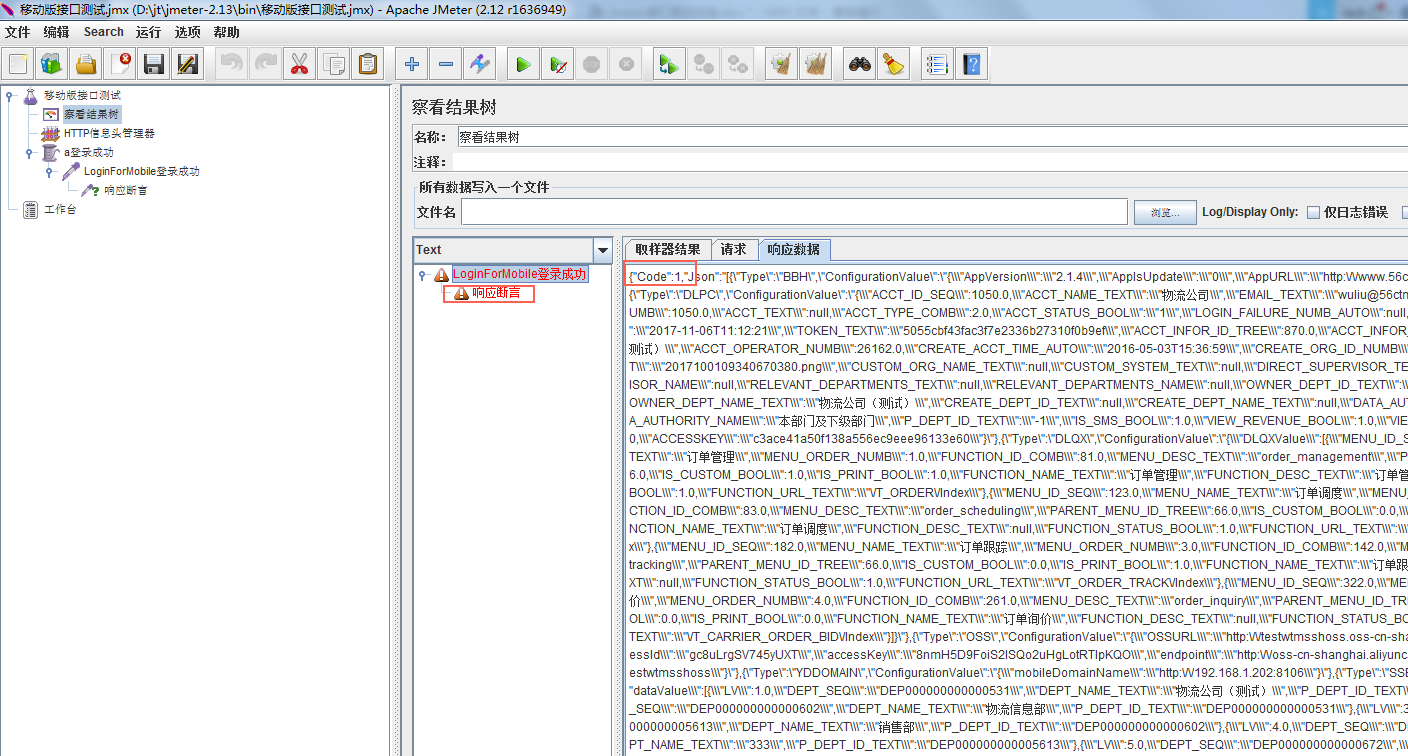


填写预期的返回结果 例如判断返回数据是否为Code":2，再运行察看结果树



&查看断言结果

因为登录的结果是正确的返回code 1,设置的断言是code 2所以报错显示红色



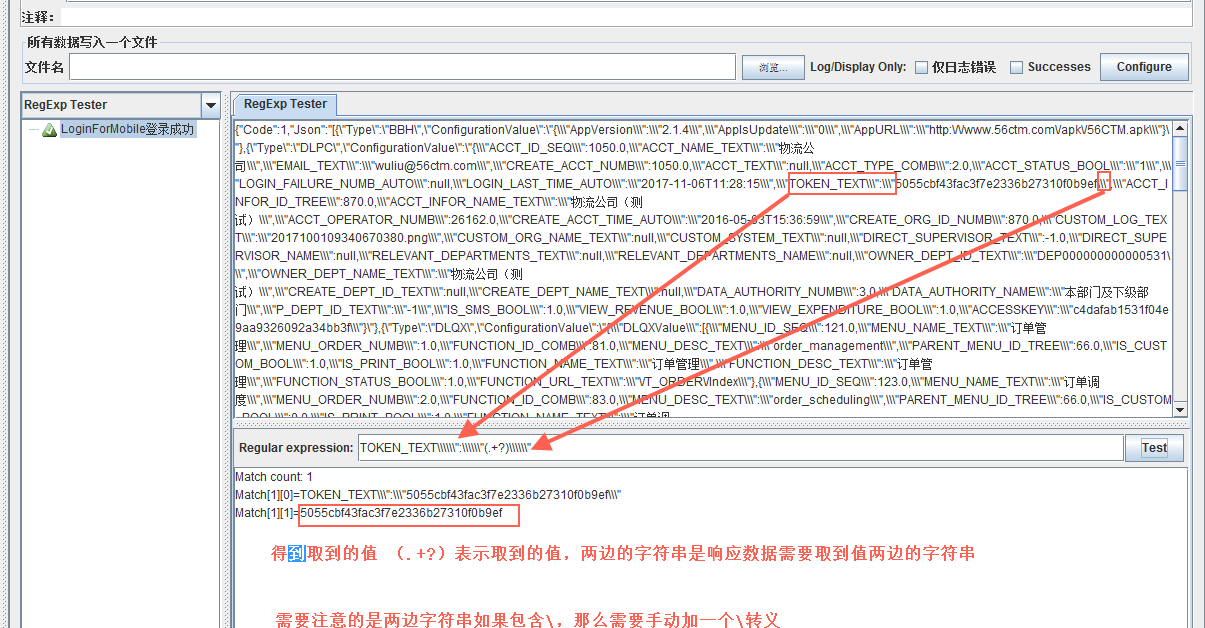
#### 4.9正则提取器

登录后系统其他接口请求时都需要token或accessk等的一个签名认证，需要用正则取出登录接口返回数据里token和accesskey传递到其他接口

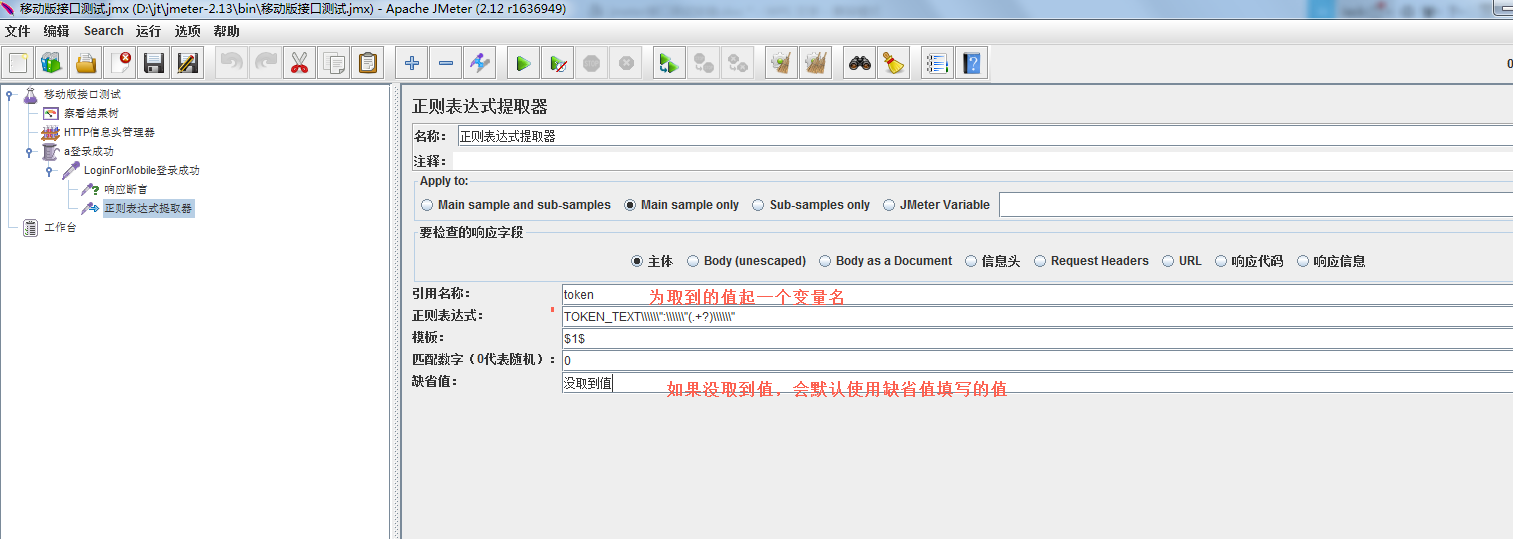
**&**正则简单运用取值

组件：【http请求】--右击添加--【后置处理器】--【正则提取器】

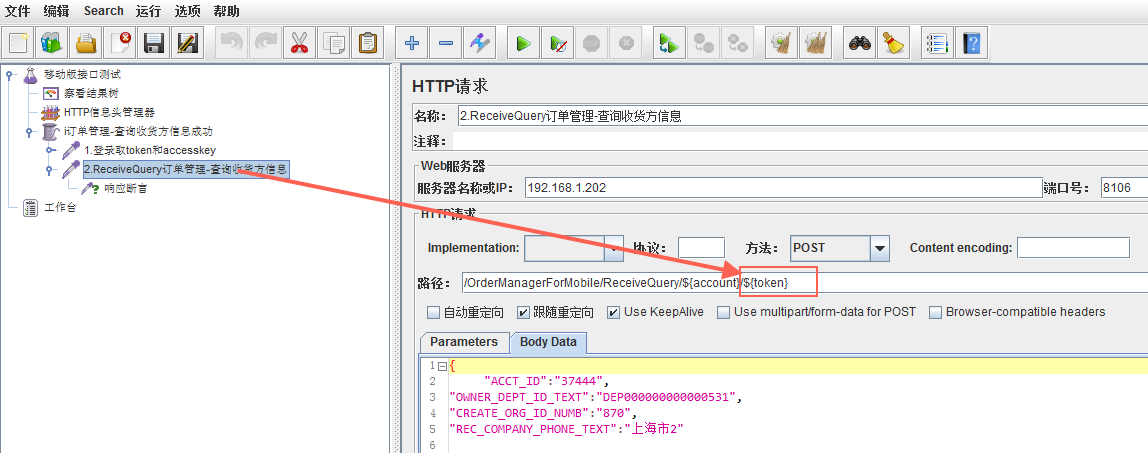
正则简单运用和测试是否取到token、accesskey方法一致（这个图务必放大看下）



写好的正则表达式放到正则提取器上填写相关信息（如图）

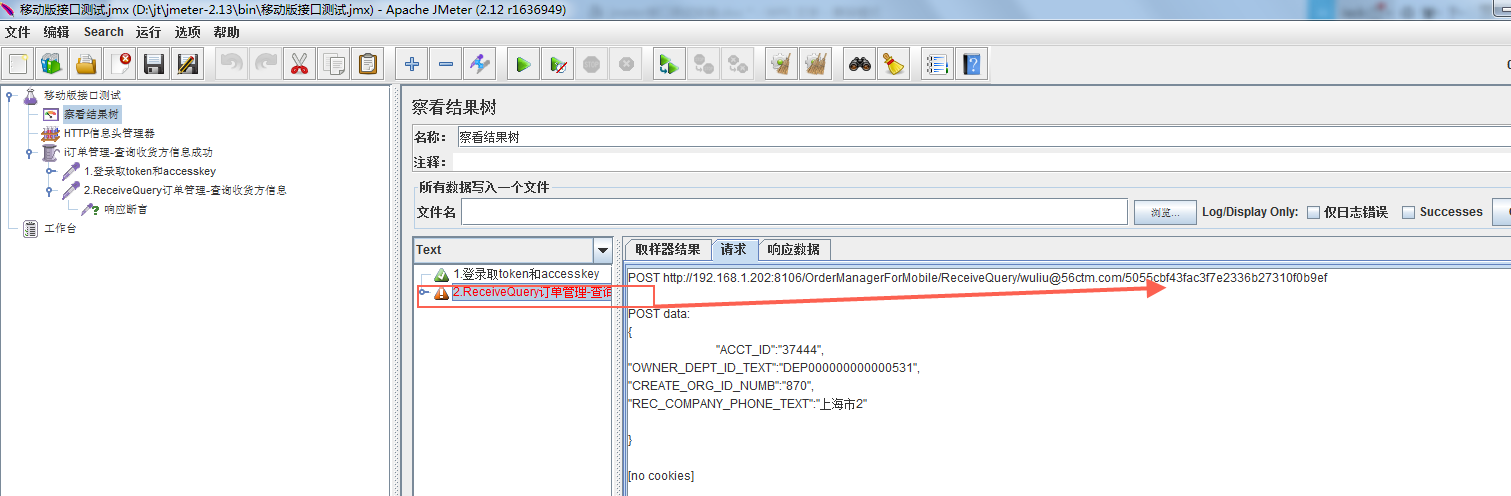


下一个接口引用变量的方式



& 查看正则取值结果

点击运行查看这个接口的token是否用正则写成功，引用的变量自动取值了（注：这个接口失败因为需要accesskey，这里只做示范所以没加accesskey,取值传递方法一致）

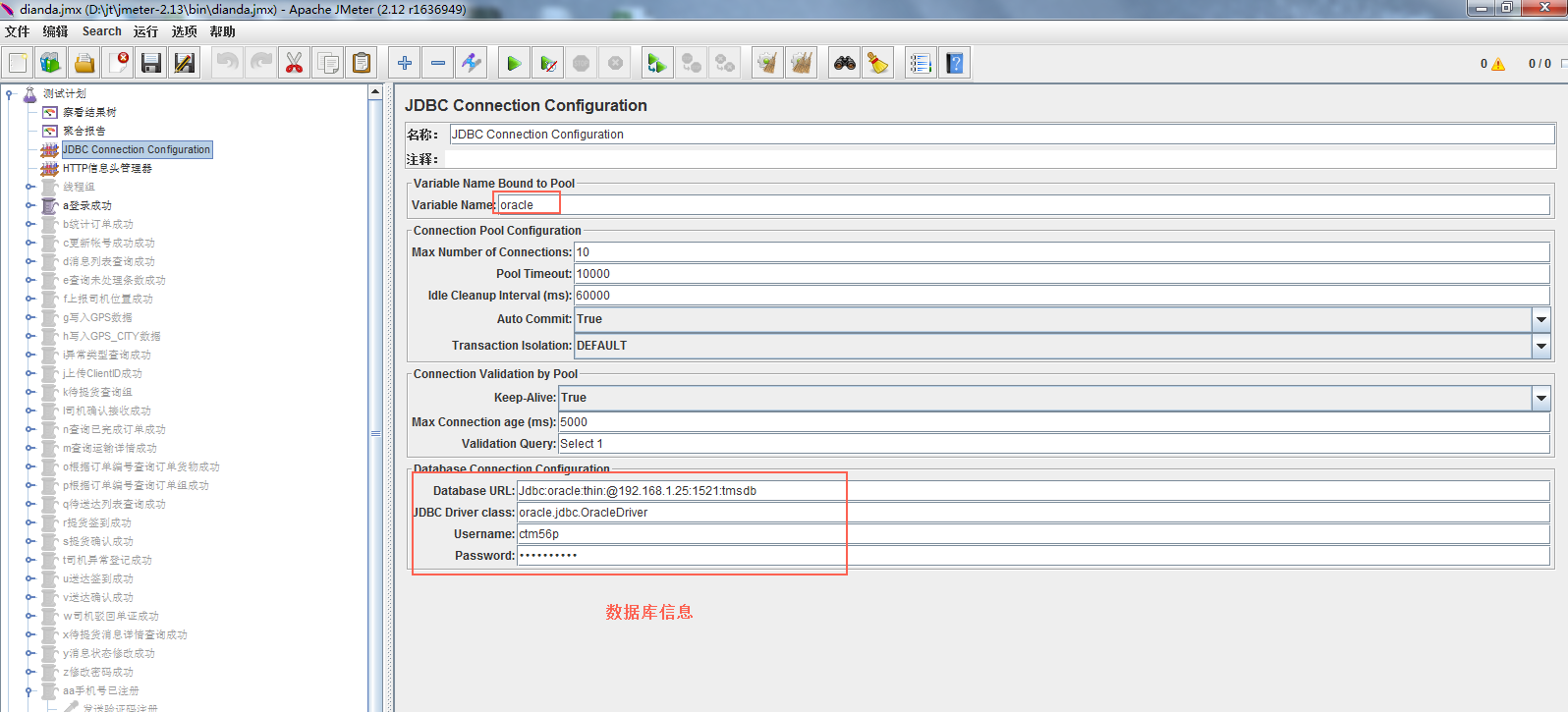


#### 5.0JDBC数据库取值传递参数

应用场景例如：注册、重置密码等接口需要从数据库把验证码取出来放到接口上发送请求

组件：jdbc配置原件 【项目名】--【右击添加】--【配置元件】--【JDBC Connection Configuration】

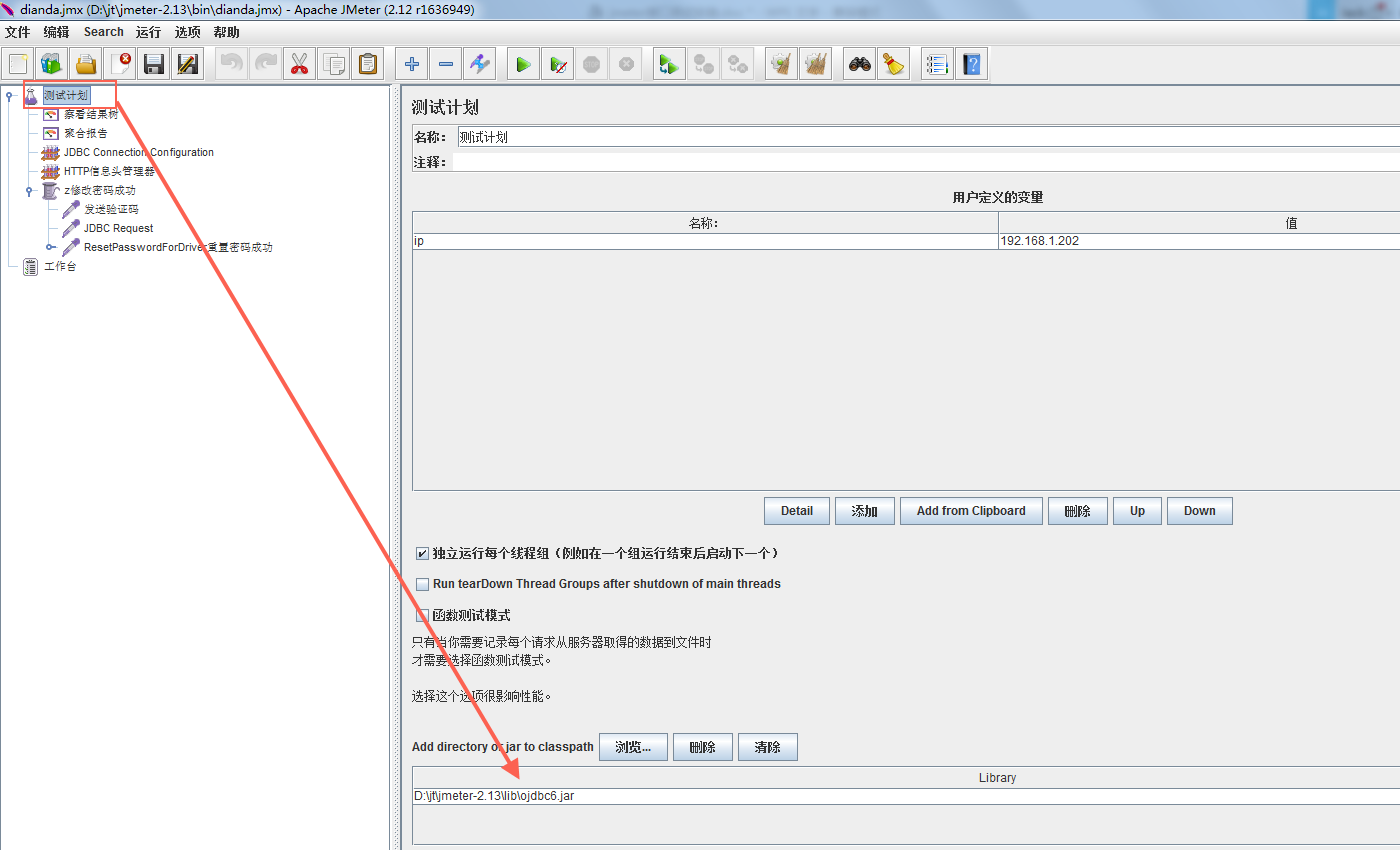
配置数据库（如图，针对oracle）



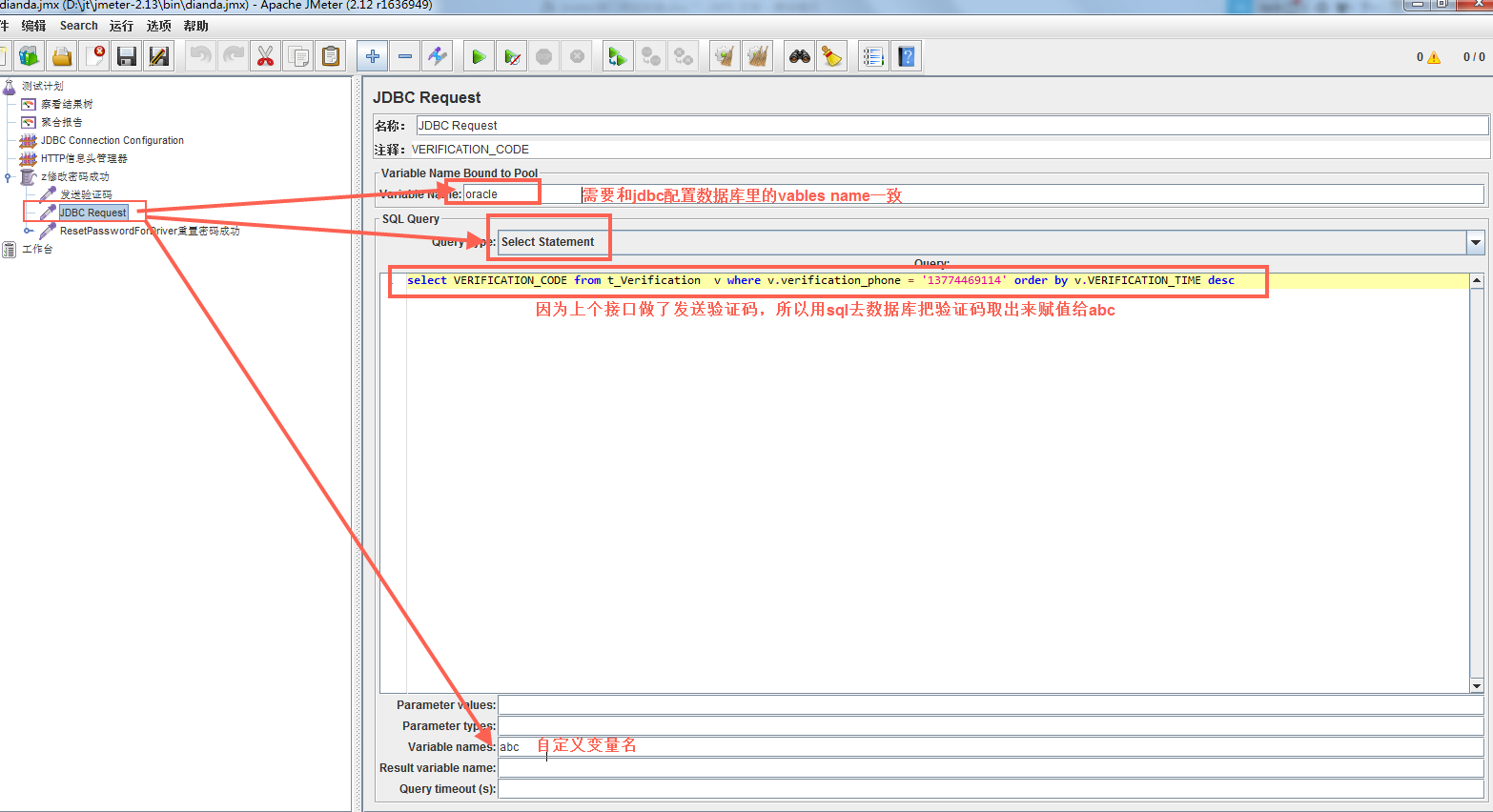
**&**数据库取值传递

重置密码接口例子@

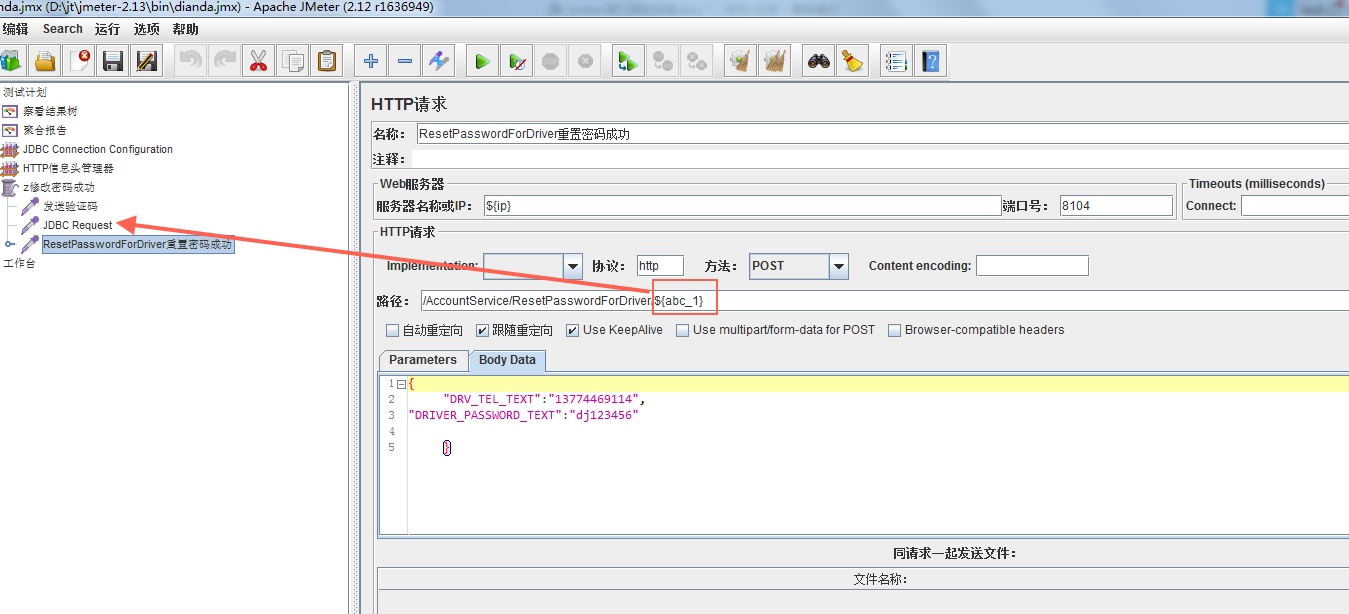
&加载ojdbc6.jar包



& 添加一个jdbc requset组件 【线程组】--添加--【Sampler】--【jdbc request】



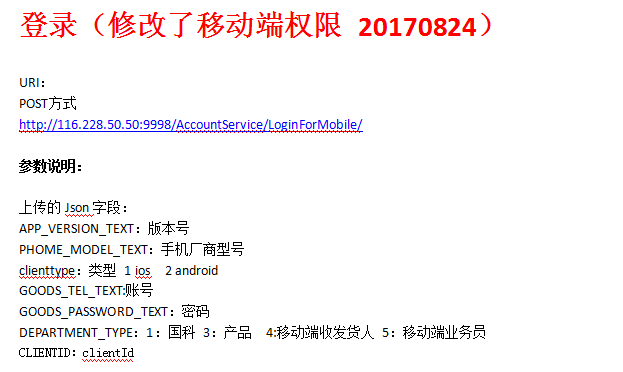
& 再做重置密码接口把上一步取的值赋值给重置密码接口数据库取值引用方式为 ${abc\_1}



#### 6.1post上传json格式和list写法

**&**正常的上传json格式

接口文档：



根据参数说明编写如下格式：

{

"GOODS\_TEL\_TEXT":"13774469114",

"DEPARTMENT\_TYPE":"5",

"GOODS\_PASSWORD\_TEXT":"dj123456",

"CLIENTID":"1",

"APP\_VERSION\_TEXT":"1",

"PHOME\_MODEL\_TEXT":"huawei",

"IP\_ADDRESS":"1",

"clienttype":"2"

}

&list形式的json

字段写在list中，接口文档如下：



根据上传json字段，编写Json格式如下

{

“dataValue”:”[{ord\_id\_seq:123,acct\_id:854}]”

}