Robot Framework 自动化测试

作者：虫师

Robot Framework 特点：

- 使用简单
- 非常丰富的库
- 可以像编程一样写测试用例
- 支持开发系统关键字

目录

第 1 章 Robot Framework 介绍

本章对 Robot FrameWork 进行介绍。

1.1 介绍

Robot Framework 的架构是一个通用的验收测试和验收测试驱动开发的自动化测试框架（ATDD）。它具有易于使用的表格来组织测试过程和测试数据。

New Test Case		
open browser	http://www.baidu.com	
input text	id=kw	robot framework
click button	id=su	
close browser		

它使用关键字驱动的测试方法。

对于上面的例子来说，open browser 、input text、click button 和 close browser，都是“关键字”，这些关键字由 robotframework-selenium2library 类库所提供。当然，我们也可以自定义关键字。

其检测能力可以通过测试库实现可以使用 Python 或 Java 的扩展，用户可以使用相同的语法，用于创建测试用例创建新的更高层次的现有的关键词。

Robot Framework 的操作系统和应用独立框架。核心框架是使用 Python 和运行在 Jython（JVM）和 IronPython（.NET）。

1.2 特点

Clear

Robot Framework has a modular architecture that can be extended with bundled and self-made test libraries.

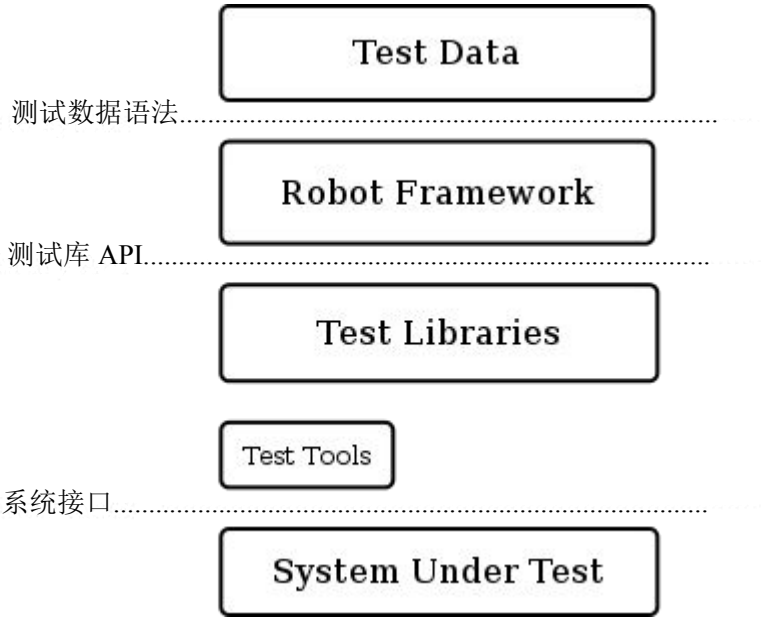
Test data is defined in files using the syntax shown in the examples below. A file containing test cases creates a test suite and placing these files into directories creates a nested structure of test suites.

Easy

When test execution is started, the framework first parses the test data. It then utilizes keywords provided by the test libraries to interact with the system under test. Libraries can communicate with the system either directly or using other test tools as drivers.

Test execution is started from the command line. As a result you get report and log in HTML format as well as an XML output. These provide extensive look into what your system does.

Modular.



1.3 例子

下面通过官方网站的一个例子，从感官上认识一下通过 Robot Framework 是如何编写测试脚本的。

```
*** Settings ***
Test Template       Calculate
Library             CalculatorLibrary

*** Test Cases ***      Expression      Expected
Additions               12 + 2 + 2      16
                        2 + -3          -1

Substractions           12 - 2 - 2      8
                        2 - -3          5

Multiplication          12 * 2 * 2      48
                        2 * -3          -6

Division                12 / 2 / 2      3
                        2 / -3          -1

Calculation error       [Template]      Calculation should fail
                        kekkonen        Invalid button 'k'.
                        ${EMPTY}        Invalid expression.
                        1 / 0           Division by zero.

*** Keywords ***
Calculate
    [Arguments]     ${expression}       ${expected}
    Push buttons    C${expression}=
    Result should be    ${expected}

Calculation should fail
    [Arguments]     ${expression}       ${expected}
    ${error} =      Should fail     C${expression}=
    Should be equal     ${expected}     ${error}
```

简单来分析一下 Robot Framework 编写测试脚本的套路：

Settings

用于导入相关的库（Library），就像在通过 Python 编写脚本时，先 import 相关的模块一样。如 CalculatorLibrary 就是导入的库。

Test Cases

用于编写测试用例，在编写用例的过程中需要使用 Library 中所提供的关键字（或使用自定义的关键字）。在编程语言中可以看作调用模块所提供的类或方法。如，Additions、Substractions 等就是 Library 所提供的关键字。

*** Keywords ***

用于自定义关键字。在编程语言中就是可以理解成自己编写的函数，类、方法等。如，Calculate、Calculation should fail 就是自定义关键字。

1.4 所支持的测试库

不同的测试库完成不同的测试功能，Robot Framework 通过导入不同的库，就可以使用库中所提供的关键字，从而时行相关的测试。

有几个标准库是和 Robot Framework 捆绑在一起，除此之外 Robot Framework 还有大量的被分别开发的外部库，你可以根据需要安装。当然，你也可以创建自己的测试库。

下面是 Robot Framework 官方网站所提供的库。

标准库：

库	介绍
Builtin	Provides a set of often needed generic keywords. Always automatically available without imports.
Dialogs	Provides means for pausing the test execution and getting input from users.
Collections	Provides a set of keywords for handling Python lists and dictionaries.
OperatingSystem	Enables various operating system related tasks to be performed in the system where Robot Framework is running.
Remote	Special library acting as a proxy between Robot Framework and test libraries elsewhere. Actual test libraries can be running on different machines and be implemented using any programming language supporting XML-RPC protocol.
Screenshot	Provides keywords to capture screenshots of the desktop.

String	Library for generating, modifying and verifying strings.
Telnet	Makes it possible to connect to Telnet servers and execute commands on the opened connections.
XML	Library for generating, modifying and verifying XML files.
Process	Library for running processes in the system. New in Robot Framework 2.8.
DateTime	Library for date and time conversions. New in Robot Framework 2.8.5.

外部库：

库	介绍
Android library	Library for all your Android automation needs. It uses Calabash Android internally.
AnywhereLibrary	Library for testing Single-Page Apps (SPA). Uses Selenium Webdriver and Appium internally.
AppiumLibrary	Library for Android- and iOS-testing. It uses Appium internally.
Archive library	Library for handling zip- and tar-archives.
AutoItLibrary	Windows GUI testing library that uses AutoIt freeware tool as a driver.
Database Library (Java)	Java-based library for database testing. Works only with Jython.
Database Library (Python)	Python based library for database testing. Works with any Python interpreter, including Jython.
Diff Library	Library to diff two files together.
Eclipse Library	Library for testing Eclipse RCP applications using SWT widgets.
robotframework-faker	Library for Faker, a fake test data generator.
FTP library	Library for testing and using FTP server with Robot Framework.
HTTP library (livetest)	Library for HTTP level testing using livetest tool internally.
HTTP library (Requests)	Library for HTTP level testing using Request internally.

iOS library	Library for all your iOS automation needs. It uses Calabash iOS Server internally.
MongoDB library	Library for interacting with MongoDB from RobotFramework using pymongo.
Rammbock	Generic network protocol test library that offers easy way to specify network packets and inspect the results of sent and received packets.
RemoteSwingLibrary	Library for testing and connecting to a java process and using SwingLibrary, especially Java Web Start applications.
SeleniumLibrary	Web testing library that uses popular Selenium tool internally. Uses deprecated Selenium 1.0 and thus not recommended for new projects.
Selenium2Library	Web testing library that uses Selenium 2. For most parts drop-in-replacement for old SeleniumLibrary.
Selenium2Library for Java	Java port of the Selenium2Library.
SSHLibrary	Enables executing commands on remote machines over an SSH connection. Also supports transfering files using SFTP.
SudsLibrary	A library for functional testing of SOAP-based web services based on Suds, a dynamic SOAP 1.1 client.
SwingLibrary	Library for testing Java applications with Swing GUI.
watir-robot	Web testing library that uses Watir tool.

通过上面的列表了解到 Robot Framework 所支持的测试非常丰富。

web 自动化测试：SeleniumLibrary，Selenium2Library，Selenium2Library for Java、watir-robot 等。

Windows GUI 测试：AutoItLibrary。

移动测试：Android library、iOS library、AppiumLibrary 等。

数据库测试：Database Library (Java)、Database Library (Python)、MongoDB library 等。

文件对比测试：Diff Library。

HTTP 测试：HTTP library (livetest)、HTTP library (Requests)等。

第 2 章 Robot Framework 环境搭建

2.1 安装 Python

访问 Python 官方网站：https://www.python.org/

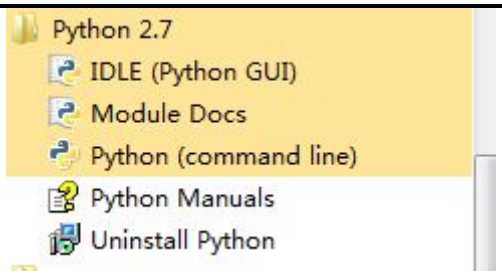
由于 Robot Framework 框架是基于 Python 语言开发的，要想使用 Robot Framework 首先需要有 Python 环境。

在学习和使用 Python 时，首先会碰到 Python 版本的问题，这在其它编程语言中是不存在的，目前 Python 分为 Python2 和 Python3 两个版本。之所以会有两个版本并存的情况，是因为随着近几年 Python 语言的逐渐流行起来，早期的 Python 为版本在基础的设计存在着一些不足之处，Python3 在设计的时候很好的解决了这些遗留问题，并且在性能上也有了很大的提升，但同时带来了新的问题就是不完全向后兼容，所以就造成了两个版本并存的情况。

由于 Robot Framework 框架是基于 Python2 开发，所以这里我们选择安装 Python2。

下载最新版本的 Python2，截止作者发稿，最新版本为 Python2.7.8 版本。读者根据自己的平台选择相应的版本进行下载；对于 Windows 用户来说，如果你的系统是 32 位的请选择 x86 版本，如果是 64 位系统请选择 64 版本进行下载。下载完成会得到一个以.msi 为后缀名的文件，双击进行安装。如图 2.1。

安装过程与其它 Windows 程序一样，安装完成在开始菜单中将看到安装好的 Python 目录：



2.2 安装 steuptools 与 pip

setuptools 和 pip 并非必须安装的两个包，但安装之后，后续再安装 Python 的库将变得非常简单。所以这里建议安装。

setuptools 是 Python Enterprise Application Kit（PEAK）的一个副项目，它是一组 Python 的 distutilsde 工具的增强工具可以让程序员更方便的创建和发布 Python 包，特别是那些对其它包具有依赖性的状况。

经常接触 Python 的同学可能会注意到，当需要安装第三方 Python 包时，可能会用到 easy_install 命令。easy_install 是由 PEAK 开发的 setuptools 包里带的一个命令，所以使用 easy_install 实际上是在调用 setuptools 来完成安装模块的工作。

pip 是一个安装和管理 Python 包的工具，通过 pip 去安装 Python 包将变得十分简单，我们将省去了搜索--查找版本--下载--安装等繁琐的过程。pip 的安装依赖于 setuptools，所以在安装 pip 之间需要先安装 setuptools。需要注意的是目前 python3 并不支持 setuptools，需要使用 distribute。

setuptools 与 pip 下载地址：

https://pypi.python.org/pypi/setuptools

https://pypi.python.org/pypi/pip

通过上面的地址进行下载，将得到下面两个包（随着时间包的版本号会有变化）。

setuptools-7.0.zip

pip-1.5.6.tar.gz

通过解压工具进行解压将得到两个文件夹，在 Windows 命令提示符进入到文件解压目录，通过 Python 执行安装文件 setup.py 进行安装。安装 setuptools：

cmd.exe

```
C:\package\setuptools-7.0>python setup.py install
```

安装 pip 的方法与 setuptools 相同，切换到 pip 解压目录，运行 setup.py 文件：

cmd.exe

```
C:\package\pip-1.5.6>python setup.py install
```

安装完成，在 Windows 命令提示符下敲入 pip 命令：

2.3 安装 Robot Framework

下载地址:https://pypi.python.org/pypi/robotframework/2.8.7

可以通过下载 exe 程序进行安装，Robot Framework 分别提供了，win-amd64.exe 和 win32.exe 两个 windows 版本，你可以根据自己的环境下载相应的版本，双击进行安装。

如果像安装普通的 Python 程序，可以下载 tar.gz 文件，解压并运行 setup.py 文件进行安装。

cmd.exe

```
C:\robot\robotframework-2.8.7>python setup.py install
```

因为在上一小节中我们已经安装了 pip，所以通过 pip 命令安装更为方便和快捷：

cmd.exe

```
C:\Python27\Lib\site-packages>pip install robotframwork
```

2.4 安装 wxPython

下载地址:http://www.wxpython.org/download.php

wxPython 是 Python 非常有名的一个 GUI 库，因为 RIDE 是基于这个库开发的，所以这个必须安装。

在官网上找到相应的版本下载，为 exe 可执行文件，双击按钮即可。

2.4 安装 RIDE

下载地址:https://pypi.python.org/pypi/robotframework-ride

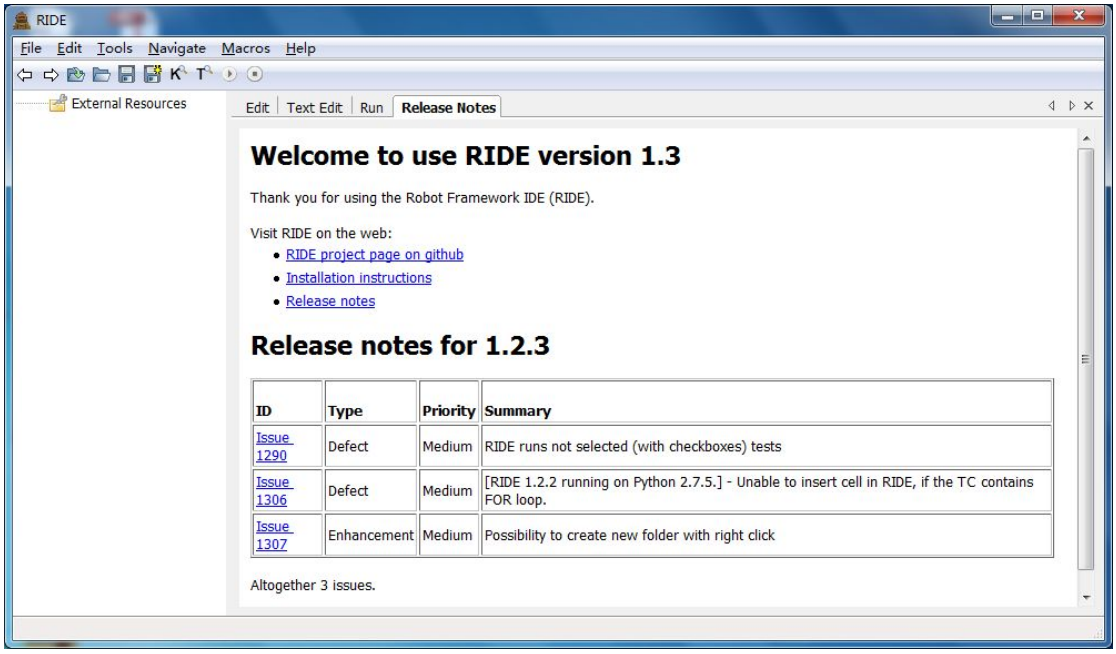
RIDE 是 Robot Framework 测试数据的编辑器。它使测试用例的创建、运行、测试项目的组织可以在图形界面下完成。

RIDE 同样提供了 win-amd64.exe 和 win32.exe 两个 windows 版本和一个 tar.gz 文件，前者下载双击进行安装。后者解压并执行 setup.py 文件。

cmd.exe

```
C:\robot\robotframework-ride-1.3>python setup.py install
```

在你安装好 RIDE 之后，桌面就会生成一个 RIDE 图标。双击启动，界面如下：



注意：

我们常常会遇到，安装完成 RIDE 后，双击桌面的 RIDE 不能启动的情况，这是由于 RIDE 所依赖的 wxPython 版本不一直造成的。

我们可以通过右击桌面 RIDE 图标--->属性：

在“目标”中会看到：“from robotide import main”的引用。打开 Python Shell 输入这行代码：

Python Shell

```
Python 2.7.8 (default, Jun 30 2014, 16:08:48) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> from robotide import main
wxPython not found.
You need to install wxPython 2.8 toolkit with unicode support to run RIDE.
wxPython 2.8.12.1 can be downloaded from
http://sourceforge.net/projects/wxpython/files/wxPython/2.8.12.1/
```

首先告诉我们没有找到 wxPython，已经了安装了的，为什么会说找不到呢？别急，接着又提示你必须安装 wxPython2.8,因为 RIDE 是基于这个版本编译的。

接着又告诉你 wxPython 2.8.12.1 版本可以在下面的地址中下载。

第 3 章 Robot Framework 入门

通过 RIDE 去学习和使用 Robot Framework 框架，对于初学者来说大大的降低了学习难度。所以后面对 Robot Framework 框架都将会通过 RIDE 中进行。实际上 RIDE 已经成为使用 Robot Framework 的“标配”。

3.1 创建项目

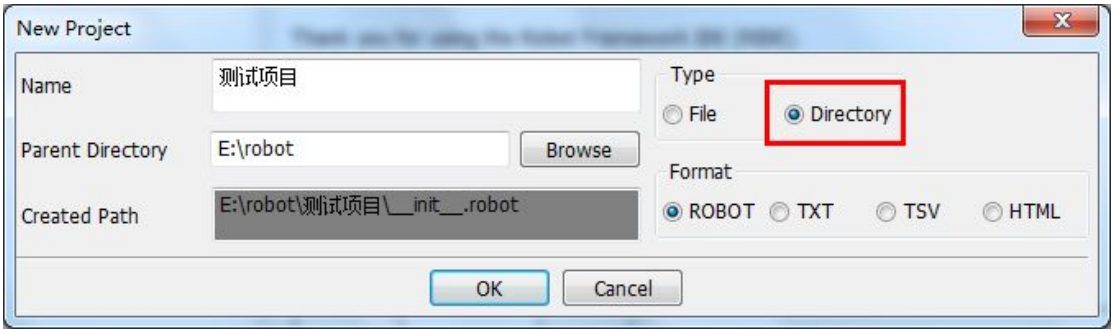
在 Robot Framework 中创建测试项目和创建单元测试项目一致。

Robot Framework	unittest(Python)
Test Project	Test Project
Test Suit	Test Suit
Test Case	Test Case

3.1.1 创建测试项目

1、创建测试项目

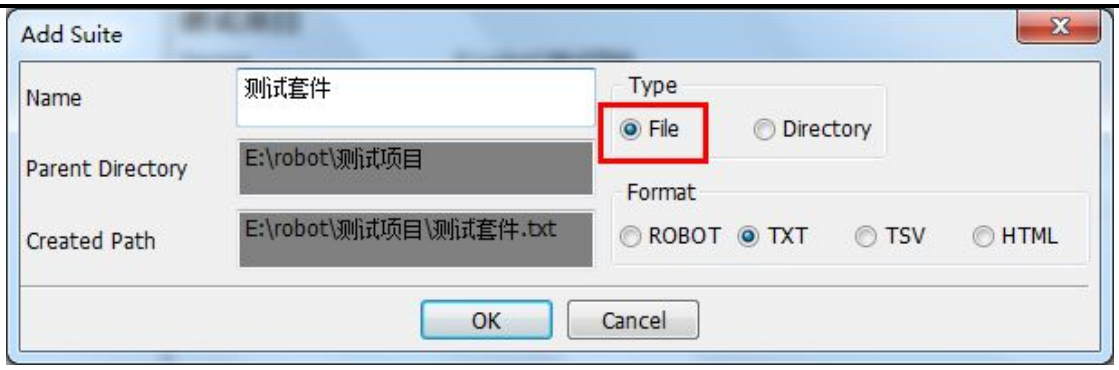
选择菜单栏 file----->new Project



Name 输入项目名称；Type 选择 Directory。

2、创建测试套件

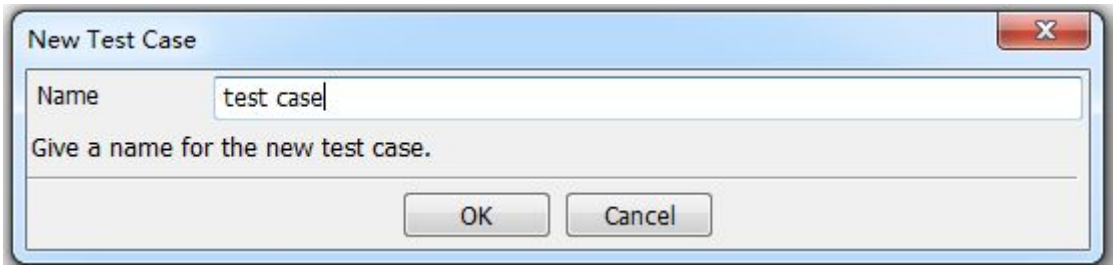
右键点击“测试项目”选择 new Suite 选项。



Name 输入项目名称；Type 选择 File。

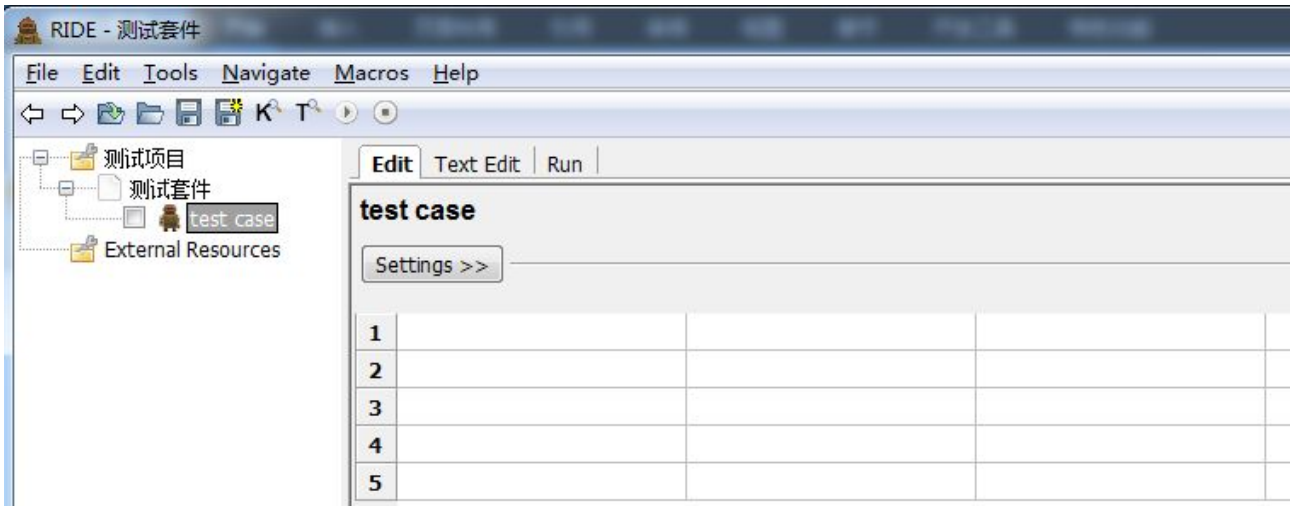
3、创建测试用例

右键点击“测试项目”选择 new Test Case。



用例只需要输入用例 name，点击 OK 即可。

完成创建之后的界面如下：

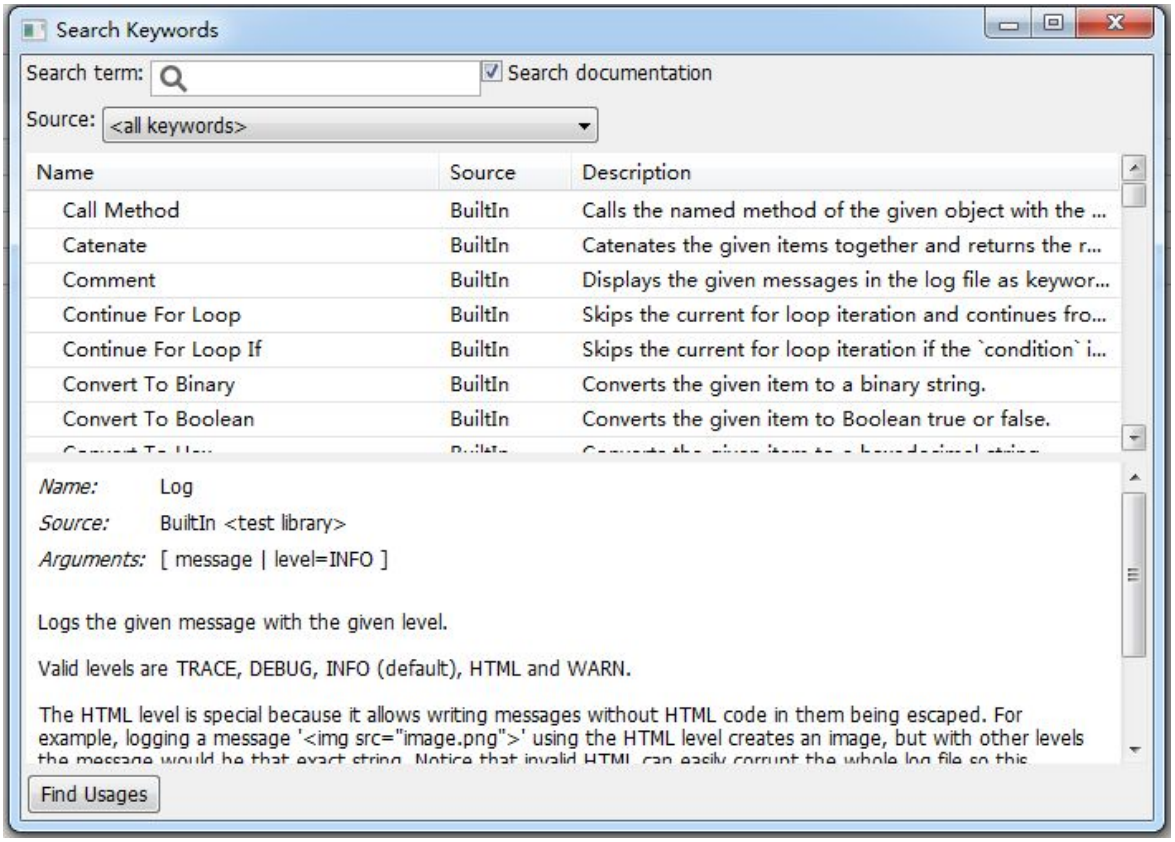


下面我们就可以在 test case 的“表格”来进行测试用例的编写了。

3.1.2 从 F5 开始学习

Robot Framework 并没有像其它框架一样提供一份完整的 API 文档，所以，我们没办法通过官方 API

文档进行习。RIDE 提供了 F5 快捷键来打开帮助文档。



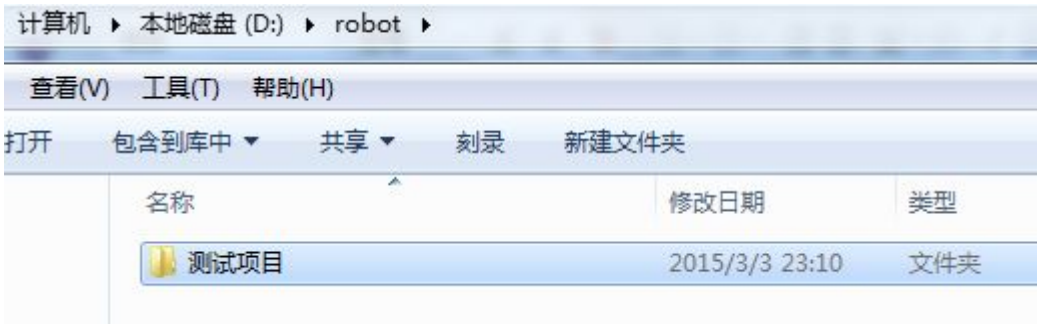
search term: 用于搜索关键字。

source：用于选择相关库，默认在所有库下搜索关键字。

创建分上下两部分，上部分显关键字列表，下半部分显示某一关键字的详细说明。你一定很好奇这些关键字的说明信息是哪儿来的？这说明是由 RIDE 读取的代码函数（方法）的系统注释获得的。在我们学到开发系统关键字的时候，将会明白这一点。

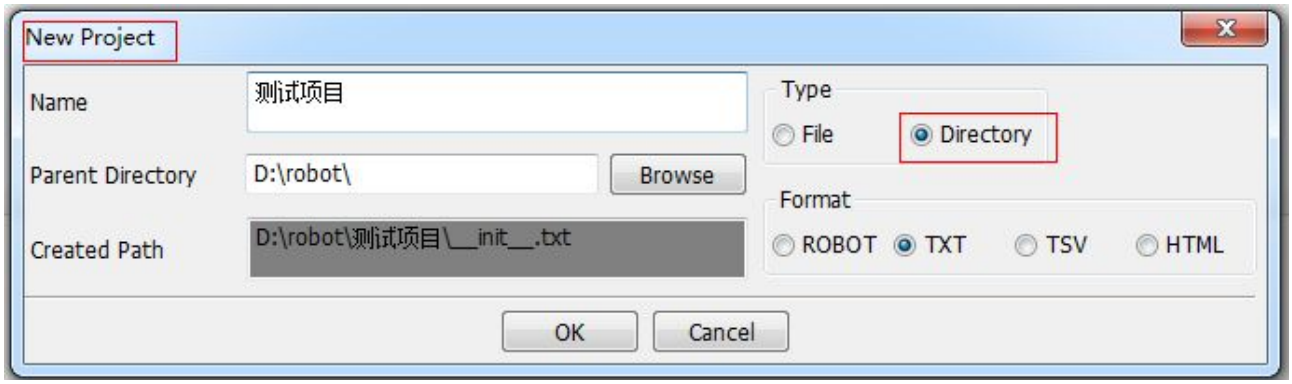
3.2 测试项目与测试套件的概念

如果你查看当前所创建的项目会发现，“测试项目”是一个目录。



“测试套件”则是一个 txt 文件。

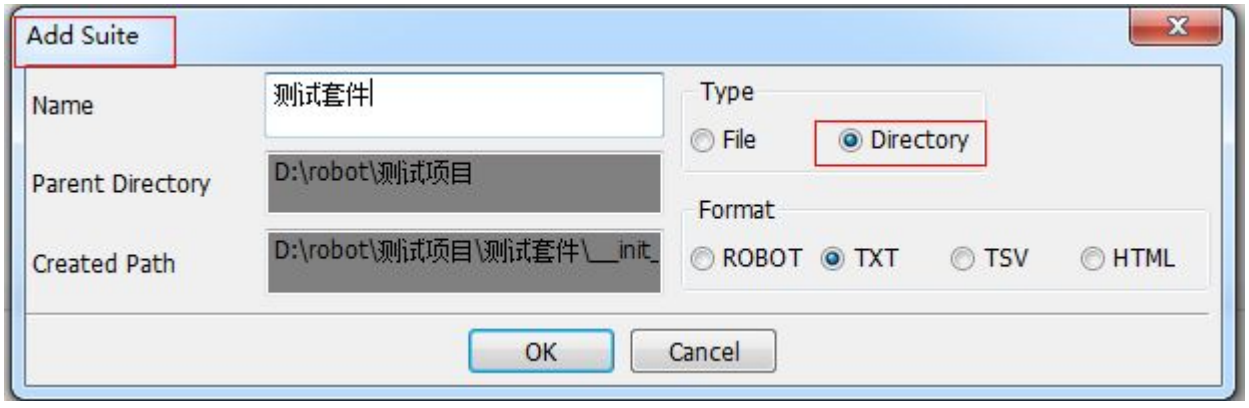
“测试项目”和“测试套件”本质上并没有什么区别，如果你愿意，也可以把测试项目创建成一个文件：



如果你把“测试项目”创建成一个文件后，那么在这个“测试项目”下就不能再创建“测试套件”了，只能创建测试用例。

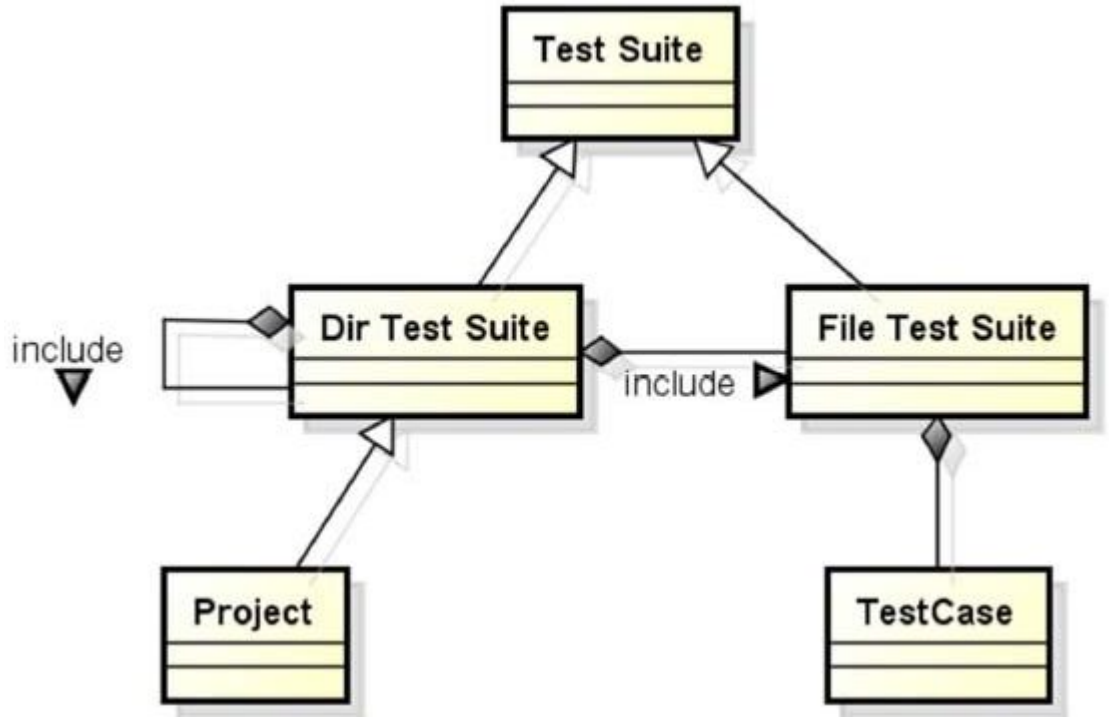
除非我们所创建的“测试项目”非常小，只需要几个用例。一般情况下，我们会选择将其定义成一个目录，这样它就可以分成多个套件，套件可定义为不同的业务，不同的业务下再分用例，结构会更加清晰。

当然，你同样也可以把“测试套件”创建成一个目录。



如果你把“测试套件”创建成了一个目录后，就不能直接在其下面创建用例了，还需要再创建的“File”类型的“子测试套件”。说白了就是用例只能创建在 file 类型的套件中。

下面用一张图来表述他们的关系：



3.3 常用关键字介绍

在学习一门编程语言的时候，大多教材都是从打印“hello world”开始。我们可以像编程语言一样来学习 Robot Framework。虽然通过 RIDE 提供“填表”一样的写测试用例的方式。但它却有着像编程语言一样的强大的关键字，以及可以开发关键字的扩展能力。

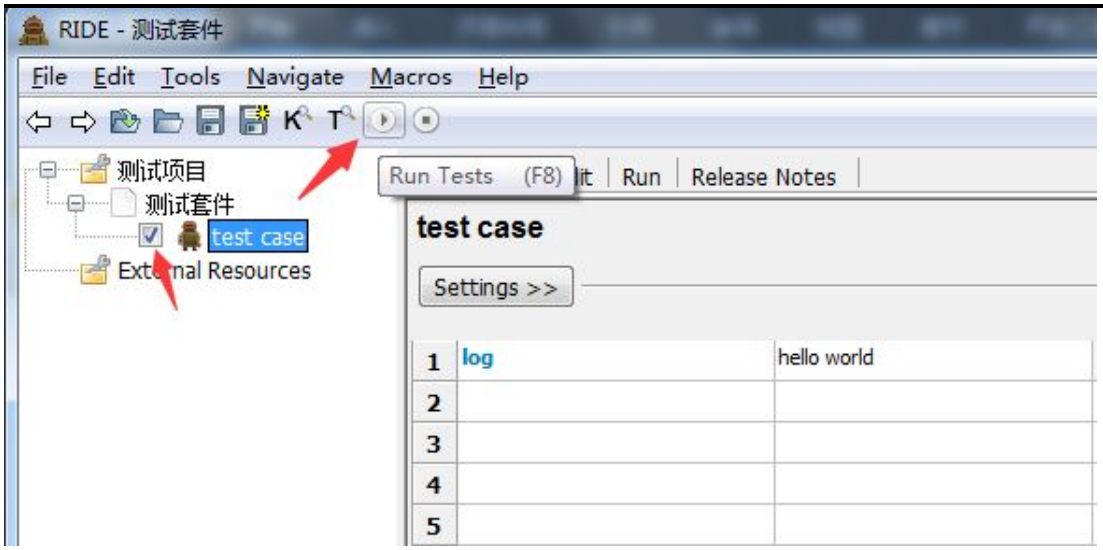
3.3.1 log 就是“print”

log 关键字就是编程语言里的“print”一样，可以打印任何你想打印的内容。

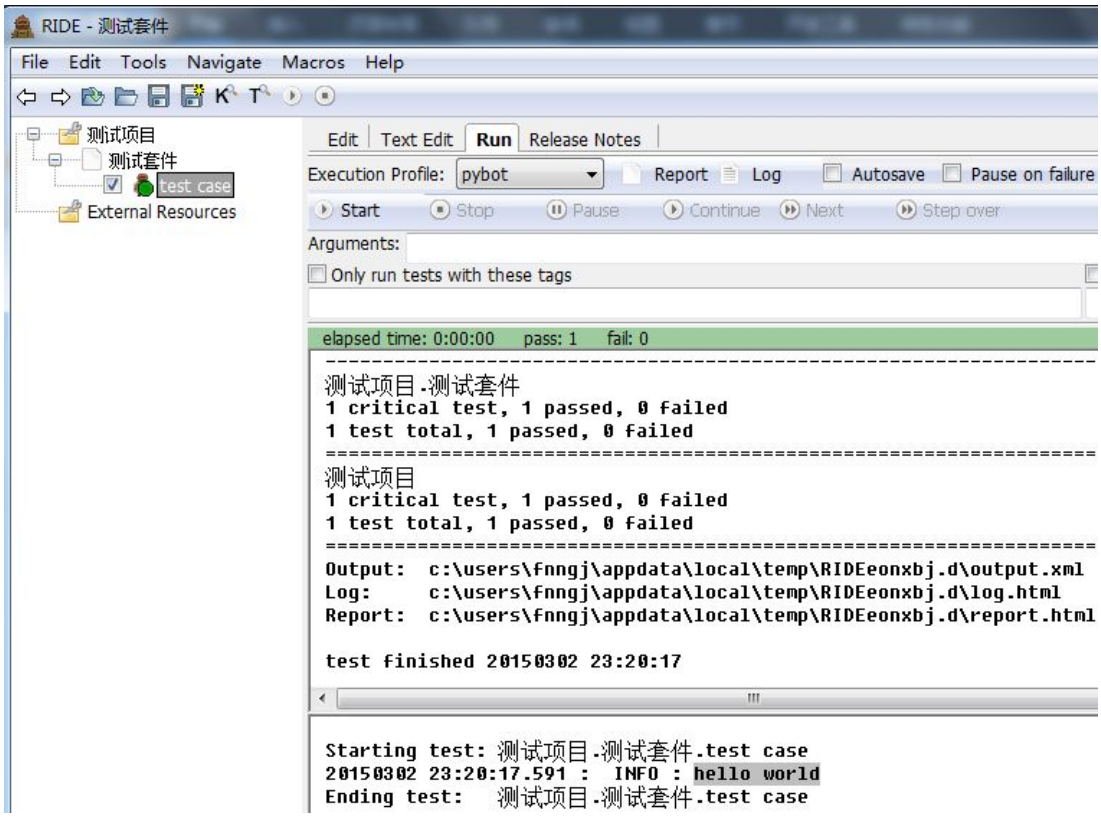
在 test case 中填写以下内容：

log	hello world	

勾选测试用例，点击工具栏的“Run Tests”按钮或按快捷键“F8”执行测试用例。如图：



在 Run 标签页，将看到执行结果：



通过运行结果中看到，“INFO:”的“hello world” 就是通过 log 关键字打印的信息。

3.3.2 定义变量

在 robot Framework 中通过“Set variable”关键字来定义变量，如：

${a}	Set variable	hello world
log	${a}	

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 09:35:33.927 :  INFO : ${a} = hello world
20150303 09:35:33.927 :  INFO : hello world
Ending test:   测试项目.测试套件.test case
```

3.3.3 连接对象

“Catenate”关键字可以连接多个信息。

${hi}	Catenate	hello	world
log	${hi}		

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 10:07:29.039 :  INFO : ${hi} = hello world
20150303 10:07:29.039 :  INFO : hello world
Ending test:   测试项目.测试套件.test case
```

加上“SEPARATOR=”可以对多个连接的信息进行分割。

${hi}	Catenate	SEPARATOR=---	hello	world
log	${hi}			

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 10:07:29.039 :  INFO : ${hi} = hello--- world
20150303 10:07:29.039 :  INFO : hello--- world
Ending test:   测试项目.测试套件.test case
```

3.3.4 定义列表

通过“Create List”关键字可以定义列表。

例 1

${abc}	Create List	a	b	c
log	${abc}			

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 10:23:20.760 :  INFO : ${a} = [u'a', u'b', u'c']
20150303 10:23:20.762 :  INFO : [u'a', u'b', u'c']
Ending test:   测试项目.测试套件.test case
```

每个字符串前面加 u，是为了统一编码问题，将字符串转为 Unicode 编码。

例 2

@{abc}	Create List	a	b	c
log many	@{abc}			

如果通过“@{}”去定义列表的话，可以通过“log many”关键字进行打印

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 17:04:30.631 :  INFO : @{abc} = [ a | b | c ]
20150303 17:04:30.632 :  INFO : a
20150303 17:04:30.633 :  INFO : b
20150303 17:04:30.633 :  INFO : c
Ending test:   测试项目.测试套件.test case
```

3.3.5 时间的操作

在 Robot Framework 中也提供操作时间的关键字。

1) Robot Framework 中提供了“get time”关键字用来获取当前时间。

${t}	get time	hello world
log	${t}	

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 17:04:30.628 :  INFO : ${t} = 2015-03-03 17:04:30
20150303 17:04:30.630 :  INFO : 2015-03-03 17:04:30
Ending test:   测试项目.测试套件.test case
```

2) “sleep”关键字用来设置休眠一定时间。

${t}	get time	hello world
sleep	5	
${t}	get time	hello world

sleep 关键字默认以“秒”为单位。

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 17:23:17.020 :  INFO : ${t} = 2015-03-03 17:23:17
20150303 17:23:22.020 :  INFO : Slept 5 seconds
20150303 17:23:22.022 :  INFO : ${t} = 2015-03-03 17:23:22
Ending test:   测试项目.测试套件.test case
```

通过前后两次获取当前时间的差，可以清楚的看到 sleep 休眠 5 秒所起的作用。

3.3.6 if 语句

通过“run keyword if”关键字可以编写 if 分支语句。

${a}	Set variable	59		
run keyword if	${a}>=90	log	优秀	
...	ELSE IF	${a}<=70	log	良好
...	ELSE IF	${a}<=60	log	及格
...	ELSE	log	不及格	

首先定义两个变量 a 等于 59 。

If 判断 a 大于等于 90 ，满足条件 log 输出 “优秀 ”；

不满足上面的条件，接着 else if 判断 a 大于等于 70 ，满足条件 log 输出 “良好”；

不满足上面的条件，接着 else if 判断 a 大于等于 60 ，满足条件 log 输出 “及格”；

上面的条件都不满足，else log 输出“不及格”。

注：注意 sele if 和 else 前面的三个点点点（...）

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 11:04:05.676 :  INFO : ${a} = 59
20150303 11:04:05.676 :  INFO : 不及格
Ending test:   测试项目.测试套件.test case
```

3.3.7 for 循环

在 Robot Framework 中编写循环通过“:for”。

例 1，执行 10 次循环。

:FOR	${i}	in range	10
	log	${i}	

通过“：for”定义 for 循环；in range 用于指定循环的范围。

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 13:24:37.713 :  INFO : 0
20150303 13:24:37.716 :  INFO : 1
20150303 13:24:37.718 :  INFO : 2
20150303 13:24:37.720 :  INFO : 3
20150303 13:24:37.723 :  INFO : 4
20150303 13:24:37.725 :  INFO : 5
20150303 13:24:37.727 :  INFO : 6
20150303 13:24:37.729 :  INFO : 7
20150303 13:24:37.732 :  INFO : 8
20150303 13:24:37.734 :  INFO : 9
Ending test:   测试项目.测试套件.test case
```

注意，in range 定义为 10，它的范围是 0~9。

例 2，遍历列表。

@{abc}	create list	a	b	c
:FOR	${i}	in	@{abc}	
	log	${i}		

“create list” 关键字用来定义列表（a,b,c），“@{}”用来存放列表。

通过过“:for”循环来来遍历@{abc}列表中的字符。

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 13:42:05.064 :  INFO : @{abc} = [ a | b | c ]
20150303 13:42:05.065 :  INFO : a
20150303 13:42:05.066 :  INFO : b
20150303 13:42:05.068 :  INFO : c
Ending test:   测试项目.测试套件.test case
```

例 3，循环中的判断

@{abc}	create list	a	b	c

:FOR	${i}	in	@{abc}	
	Exit For Loop If	'${i}'=='c'		
log	${i}			

通过“Exit For Loop If”关键字时行 for 循环内的判断，当满足 Exit For Loop If 条件后，循环结束。

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 16:24:26.146 :  INFO : @{abc} = [ a | b | c ]
20150303 16:24:26.154 :  INFO : Exiting for loop altogether.
20150303 16:24:26.158 :  INFO : c
Ending test:   测试项目.测试套件.test case
```

从执行结果看到当循环到字符 c 时，Exit For Loop If 条件成立，结束循环；通过 log 打印当前的字符 c。

3.3.8 强大的 Evaluate

为什么说“Evauate”关键字强大呢。因为通过它可以使用 Python 语言中所提供的方法。

例 1，生成随即数

在 Python 中我们可以这样来引和并使用方法：

Python Shell

```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import random
>>> random.randint(1000, 9999)
3308
```

random 模块的 randint()方法用于获取当前时间。

在 Robot Framework 中使用“Evauate”也可以调用 Python 所提供的 random 模块下的 randint()方法。

${d}	Evaluate	random.randint(1000, 9999)	random
log	${d}		

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 16:36:08.387 :  INFO : ${t} = 5031
20150303 16:36:08.389 :  INFO : 5031
Ending test:   测试项目.测试套件.test case
```

例 2，执行本地程序

Evaluate	os.system('python c:/helloworl.py')	os

通过调用 Python 所提供的 os 模块中的 system()方法可以执行本地 Python 文件。至于在.py 文件中 Python 可以做任何想做的事。

对于 system()方法来说，它也不单单可执行 Python 文件，任何在 cmd 命令提示符下可运行文件和命令，它都可以执行。

不过，一般情况下不建议通过 system()方法去执行外部程序。这样做其实就脱离了 Robot Framework，也就是说不管 Robot Framework 什么事了。我们尽量把要做的事情通过 Python 封装成关键字给 Robot Framework 使用。

3.3.9 导入库

在 Python 语言中可以使用 import 导入标准和第三方模块或框架。那么在 Robot Framework 中也提供了“Import Library”关键字来完成这个事情。

1、调用框架

Import Library	unittest	

我们导入了 Python 标准的单元测试框架 unittest。当然，在 Robot Framework 中并不能使用 unittest 单元测试框架。这只里只演示“Import Library”等同于 Python 语言中的 import。

假如我们通过“Import Library”导入一个不存在的模块，然后运行测试用例：

执行结果：

```
Starting test: 测试项目.测试套件.test case5
20150303 18:28:42.621 :  FAIL :
Importing test library 'HTMLTestRunner' failed: ImportError: No module named
HTMLTestRunner
Traceback (most recent call last):
  None
PYTHONPATH:
  C:\Python27\lib\site-packages\robot\libraries
  C:\Python27\lib\site-packages

C:\Python27\lib\site-packages\robotframework_selenium2library-1.5.0-py2.7.eg
g
  C:\Python27\lib\site-packages\setuptools-12.0.1-py2.7.egg
  C:\Python27\lib\site-packages\pip-6.0.6-py2.7.egg
  C:\Windows\system32\python27.zip
  C:\Python27\DLLs
  C:\Python27\lib
  C:\Python27\lib\plat-win
```

```
  C:\Python27\lib\lib-tk
  C:\Python27
  C:\Python27\lib\site-packages\wx-2.8-msw-unicode
  .
  D:\测试项目
Ending test:   OTC qs.测试项目.测试套件.test case5
```

Robot Framework 会遍历 Python 安装目录下的相关目录查找“HTMLTestRunner”模块。

2、调用 Python 文件

首先创建 test.py 文件。

test.py

```
#coding=utf-8

def add(a,b):
    return a+b

if __name__ == "__main__":
    a = add(4,5)
    print a
```

运行结果为 9，这是再简单不过的小程序了。

下面就通过 Robot Framework 调用 test.py 文件中的 add()函数。

Import Library	d:/test.py		
${add}	add	4	5
log	${add}		

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 21:54:21.335 :  INFO : ${add} = 45
20150303 21:54:21.335 :  INFO : 45
Ending test:   测试项目.测试套件.test case
```

从执行结果中看到和预想的结果不是样，不应该是 9 么？怎么是 45。这是因为 Robot Framework 把 4 和 5 当前两个字符串。所以 4 和 5 拼接起来是 45。我们可以将 4 和 5 转化成 int 类型，再调用 add。

Import Library	d:/test.py		
${a}	Evaluate	int(4)	
${b}	Evaluate	int(5)	

${add}	add	${a}	${b}
log	${add}		

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150303 22:01:18.430 :  INFO : ${a} = 4
20150303 22:01:18.430 :  INFO : ${b} = 5
20150303 22:01:18.430 :  INFO : ${add} = 9
20150303 22:01:18.430 :  INFO : 9
Ending test:   测试项目.测试套件.test case
```

通过“Evaluate”转化成为 int 类型后，再调用 add 就得到了想要的结果。

3.3.10 注释

Robot Framework 中添加注释也非常简单。“Comment”关键字用于设置脚本中的注释。

Comment	这是注释	

除此之外，你也可以像 Python 一样使用“#”号进行注释。

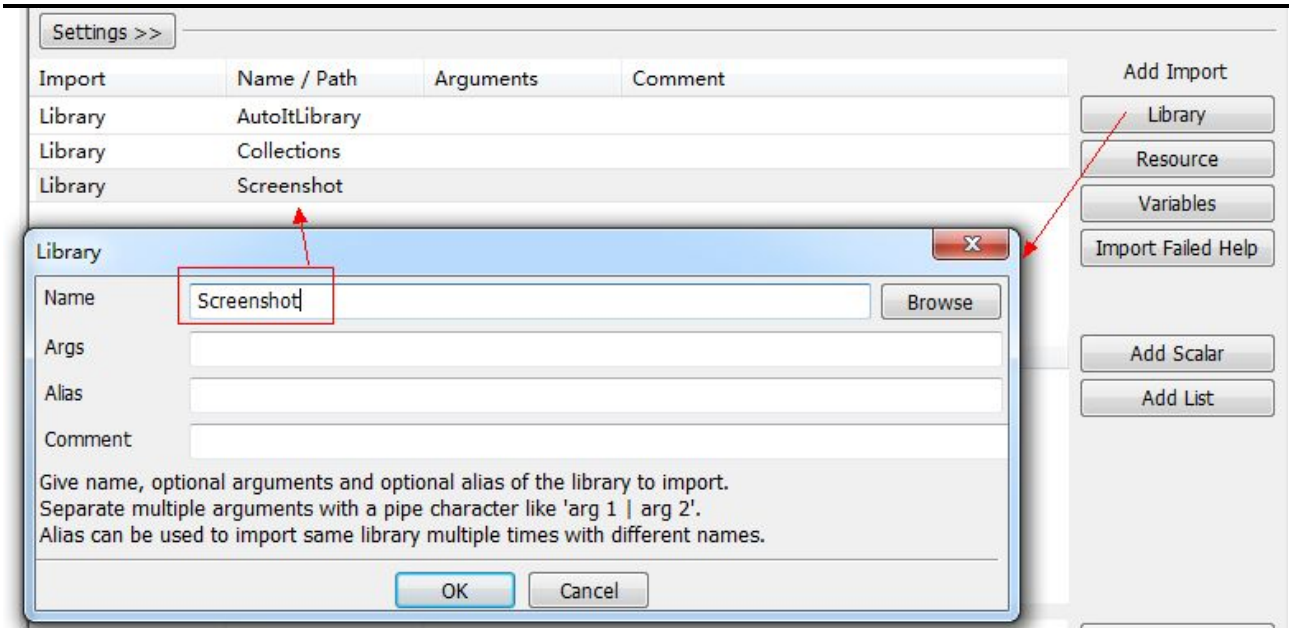
#这也是注释		

这一小节中介绍的关键字全部由 Builtin 库提供，Builtin 为 Robot Framework 标准类库。Builtin 库提供常用的关键字。

3.4 Screenshot 库

Screnshot 同样为 Robot Framework 标准类库，我们只将它提供的其它中一个关键字“Take Screenshot”，它用于截取到当前窗口。

虽然 Screnshot 也为 Robot Framework 标准类库，但它默认不会加载，需要手动加载这个库。



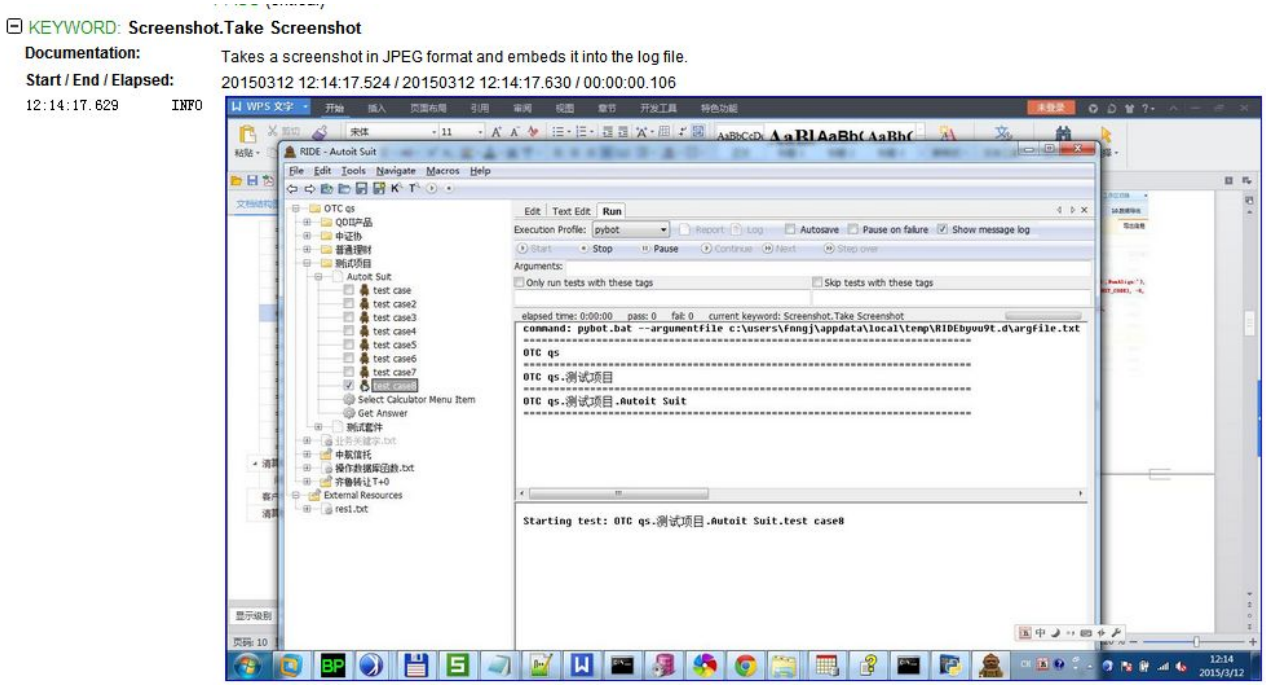
3.4.1 屏幕截图

使用“Take Screenshot”关键字实现截取当前屏幕。

例：

Take Screenshot		

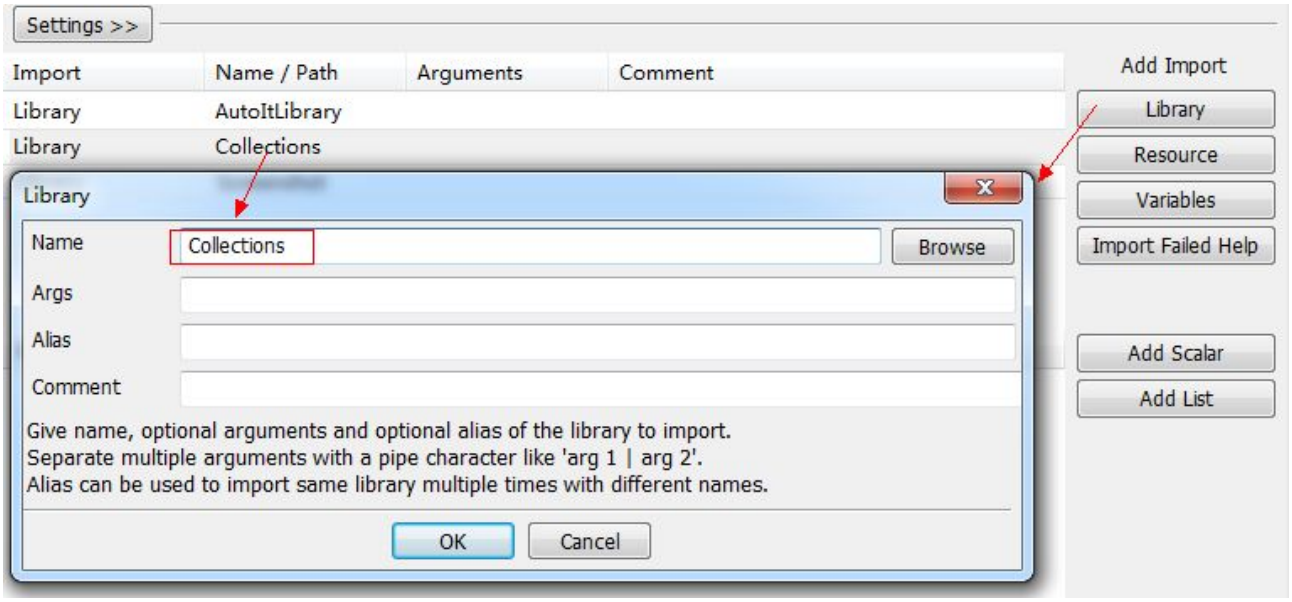
运行用例，查看 log.html：



3.5 Collections 库

Collections 库同样为 Robot Framework 标准类库，它所提供的关键字主要用于列表、索引、字典的处理。

在使用之前需要在测试套件（项目）中添加：



3.5.1 创建字典

字典也是一种常见的存放数据的形式，Create Dictionary 关键字用于创建关键字。

Create Dictionary	key	value	key	value
Create Dictionary	a	1	b	2

字典的存放方式是 key:value 成对儿存放的。

3.5.2 操作字典

Get Dictionary Items 关键字用于读取字典的 key 和 value

${dict}	Create Dictionary	a	1	b	2
${itmes}	Get Dictionary Items	${dict}			
log	${itmes}				
${key}	Get Dictionary Keys	${dict}			
log	${key}				
${value}	Get Dictionary Values	${dict}			
log	${value}				
${v}	Get From Dictionary	${dict}			
log	${v}				

Get Dictionary Items 关键字获取字典中的 key 和 value。
Get Dictionary Keys 关键字获取字典中的 key。
Get Dictionary Values 关键字获取字典中的 value。
Get From Dictionary 关键字获取字典中的 key 对应的 value

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150312 17:56:24.811 :  INFO : ${dict} = {u'a': u'1', u'b': u'2'}

20150312 17:56:24.813 :  INFO : ${itmes} = [u'a', u'1', u'b', u'2']
20150312 17:56:24.814 :  INFO : [u'a', u'1', u'b', u'2']

20150312 17:56:24.815 :  INFO : ${key} = [u'a', u'b']
20150312 17:56:24.816 :  INFO : [u'a', u'b']

20150312 17:56:24.817 :  INFO : ${value} = [u'1', u'2']
20150312 17:56:24.818 :  INFO : [u'1', u'2']

20150312 17:56:24.819 :  INFO : ${v} = 2
20150312 17:56:24.820 :  INFO : 2
Ending test:   测试项目.测试套件.test case
```

虽然 Screnshot 也为 Robot Framework 标准类库，但它默认不会加载，需要手动加载这个库。

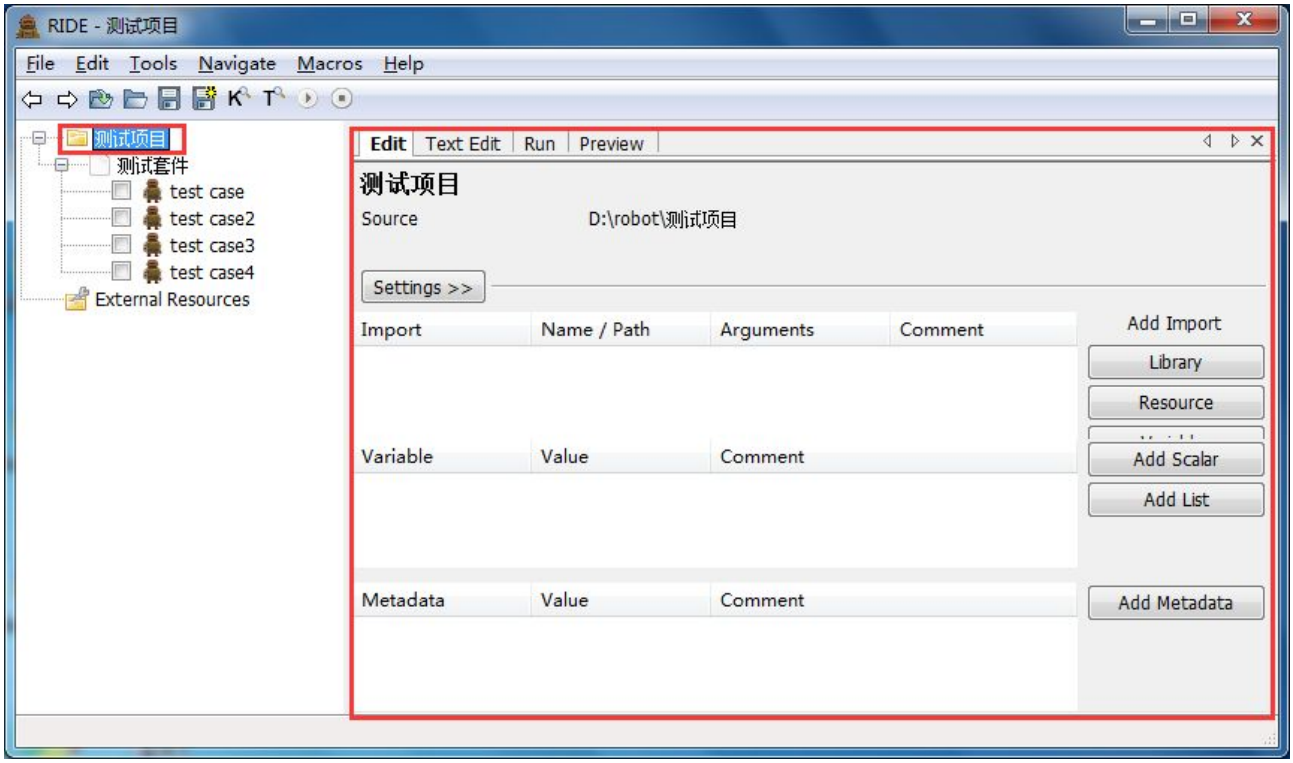
第 4 章 认识 RIDE

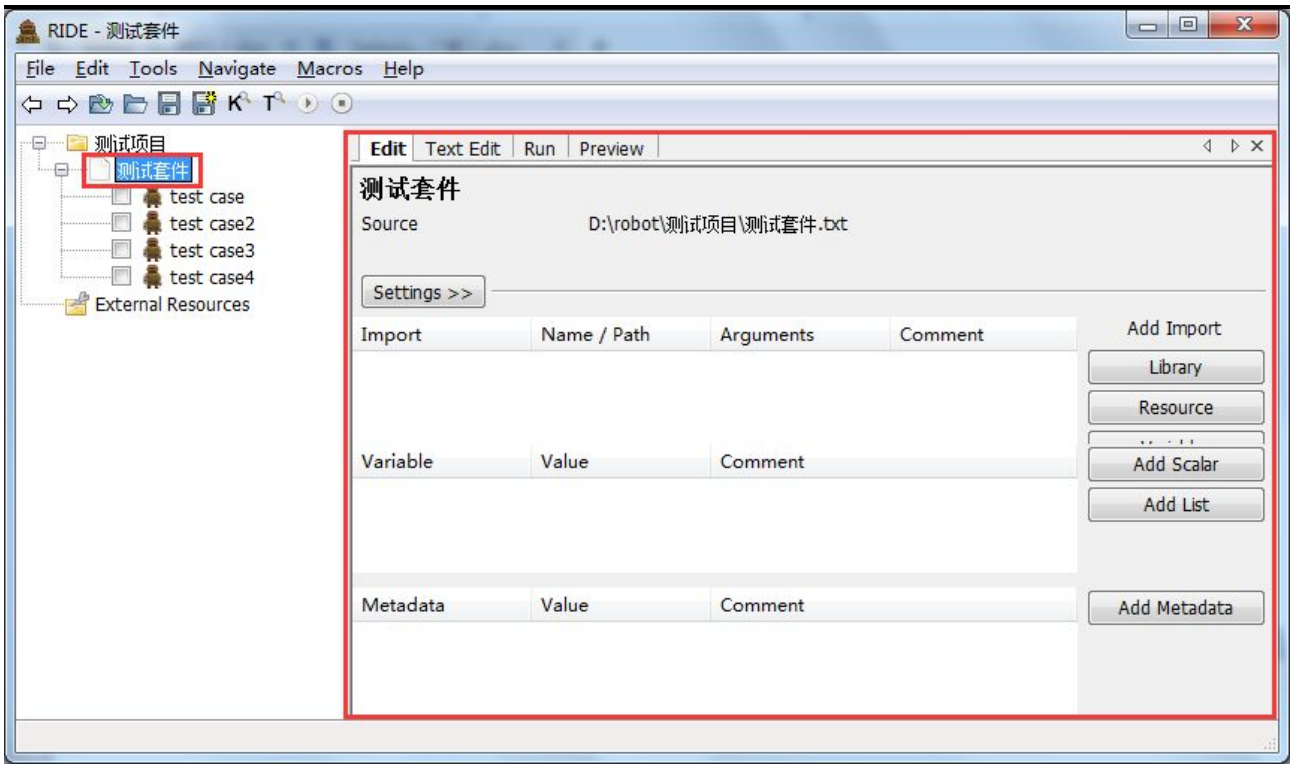
RIDE 作为 Robot Framework 的“脸面”，虽然我们已经可以拿它来创建和运行测试了，但我们对它的认识并不全面，这一小节我们将了解这个工具的使用。

4.1 Edit 标签

下面我们来看一看测试项目和测试套件所提供的 Edit 标签。

从而“测试项目”和“测试套件”所提供的 Edit 标签的功能也可看出两者是一样的。





在 Edit 标签页中主要分：加载外部文件、定义内部变量、定义元数据等三个部分。

（1）：加载外部文件

Add Library：加载测试库，主要是[PYTHON 目录]\Lib\site-packages 里的测试库

Add Resource：加载资源，主要是你工程相关的资源文件

Add Variables：加载变量文件。

（2）：定义内部变量

Add Scalar：定义变量

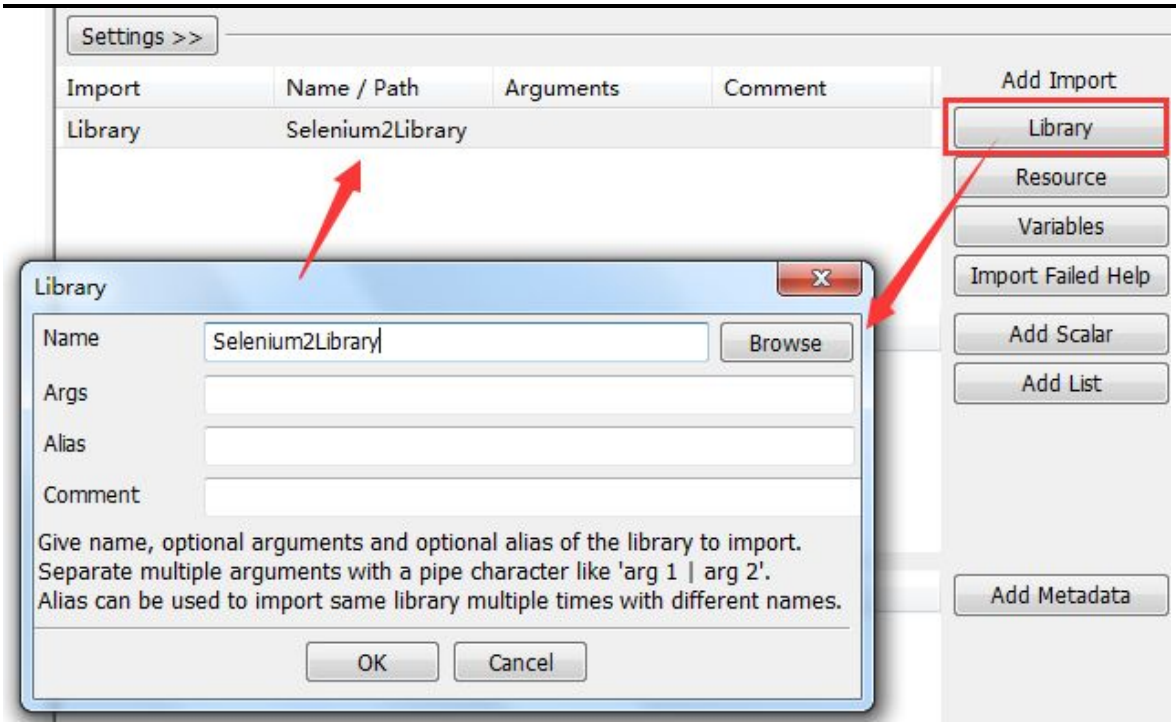
Add List：定义列表型变量

（3）：定义元数据

Add Metadata：定义元数据。我是直接翻译的，这个是新增加的部分，大概看了一下作用是在 report 和 log 里显示定义好的内容，格式和 document 一样。

4.1.1 导入库

点击 Edit 标签页右侧的“Library”按钮，来添加库。在添加库之前，首先库已经在 Python 下进行了安装。如，添加“Selenium2Library”库。

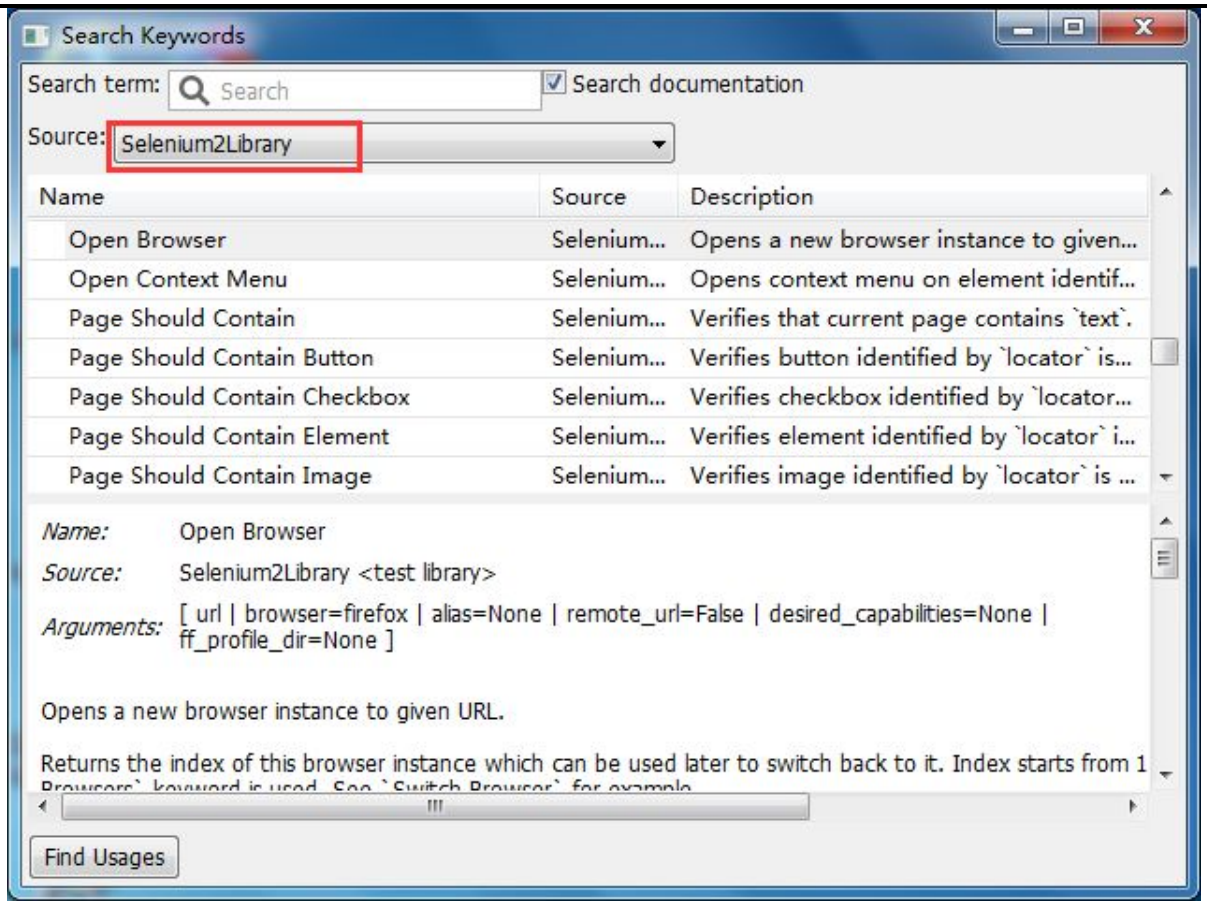


如果添加的库不存在或库名错误，将会红色显示，黑库正常表示正常。

如果你是在“测试套件”中添加的库，那么这个库中所提供的关键字可以被当前测试套件下的用例使用。

如果你是在“测试项目”中添加的库，当前项目下的测试用例不能使用库中的关键字，需要在用例相应的“测试套件”中再次添加库。

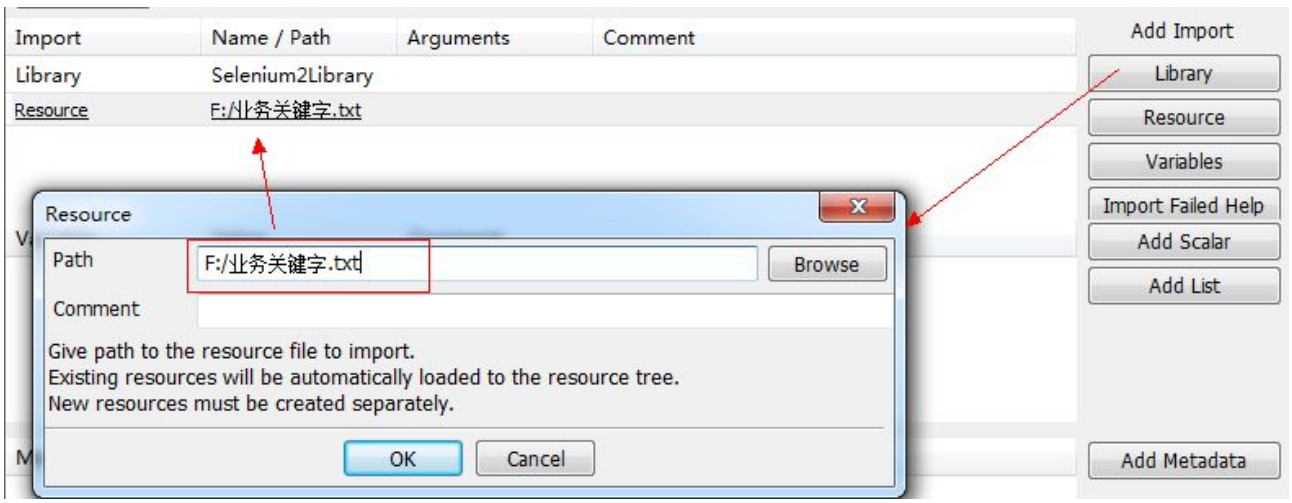
现在按 F5 就可以查看库中所提供的关键字。



4.1.2 导入资源

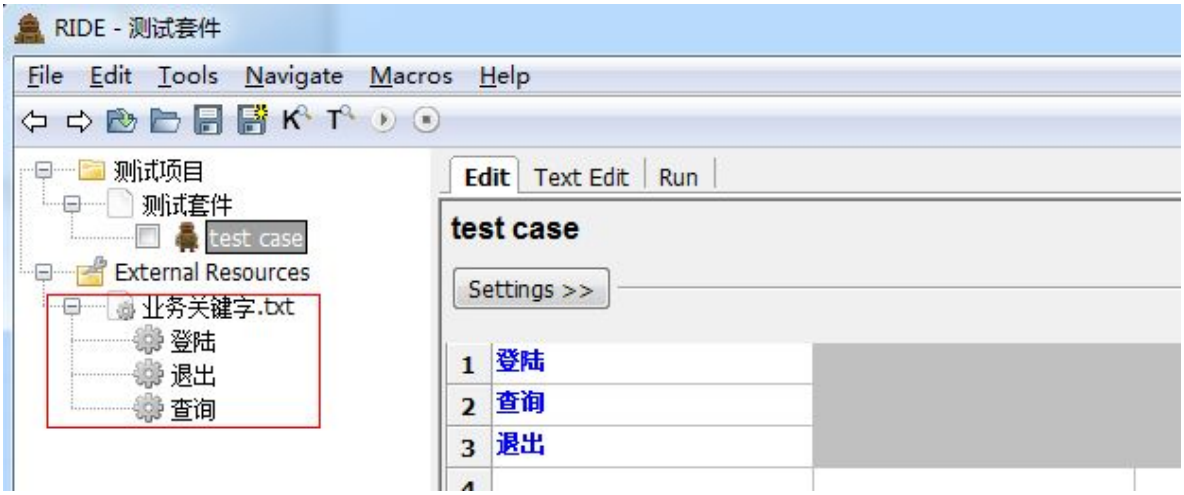
点击 Edit 标签页右侧的“Resource”按钮来添加资源。这个资源一般为项目关的文件。比如，项目的自定义关键字文件。

下面我们就来添加一个“业务关键字.txt”文件。



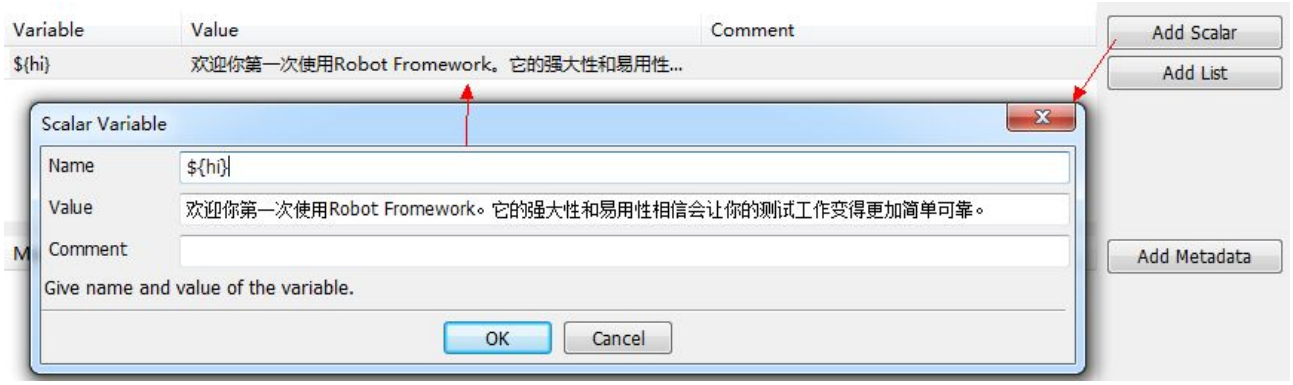
关于添加资源的作用域与库一样。我这里是添加到的测试套件中，那么它的作用域就是当前测试套件下的所有用例。

查看 RIDE 左侧项目列表，会发现“External Resource”下多了一下“业务关键字.txt”的资源。展开关键字会看到文件中定义的登录、退出和查询三个关键字。现在就可以在用例中使用这些业务关键字了。



4.1.3 定义变量

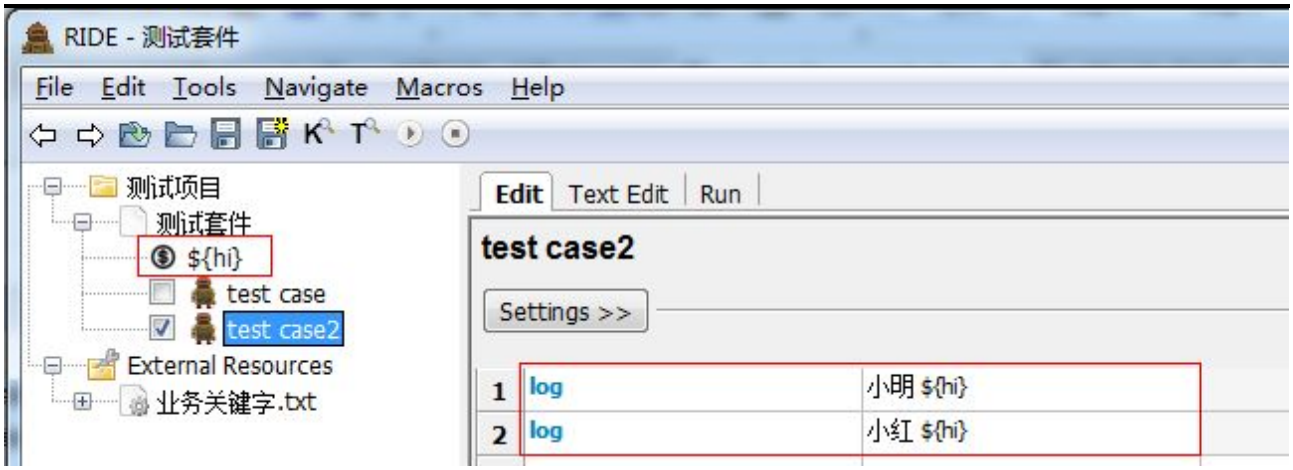
点击 Edit 标签页右侧的“Add Scalar”按钮来创建变量。这里创建的变量可以被整个测试套件中的用例所使用。也可以认为是一个“公共变量”。



Name 用于定义变量名：${hi}

Value 用于给变量赋值。这里赋值是一段话，“欢迎你第一次使用 Robot Framework。它的强大性和易用性相信会让你的测试工作变得更加简单可靠。”

下面就可以在测试用例中来使用这个变量。



用例中是两个打印信息，分别使用了${hi}变量。

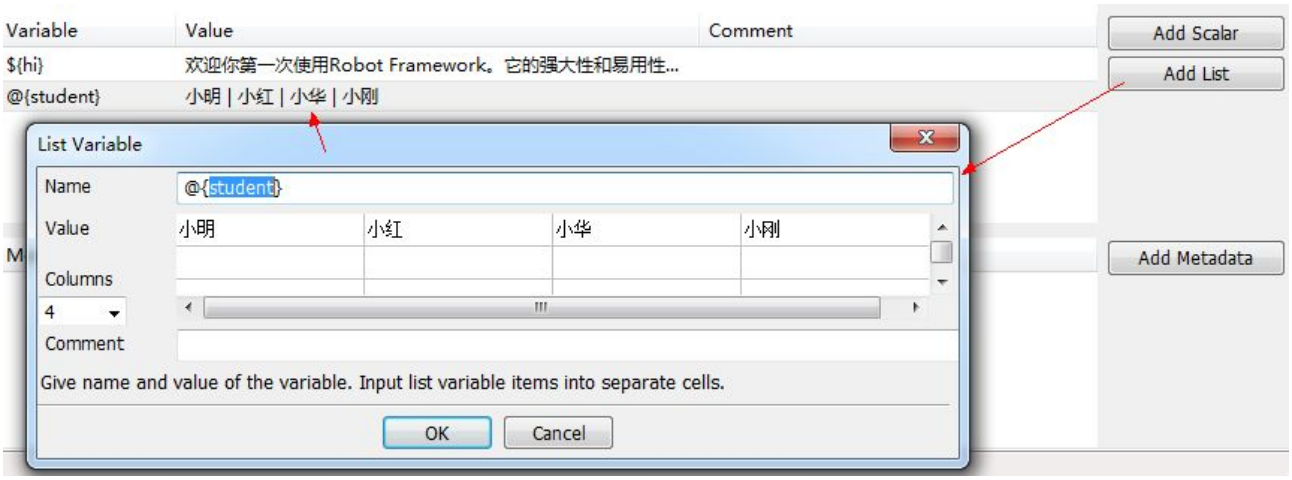
执行结果：

```
Starting test: 测试项目.测试套件.test case2
20150304 13:39:25.874 : INFO : 小明 欢迎你第一次使用 Robot Framework。它的强大性和易用性相信会让你的测试工作变得更加简单可靠。
20150304 13:39:25.875 : INFO : 小红 欢迎你第一次使用 Robot Framework。它的强大性和易用性相信会让你的测试工作变得更加简单可靠。
Ending test:  测试项目.测试套件.test case2
```

相信从用例的执行结果中我们已经体会到了“公共变量”的作用。

4.1.4 定义列表变量

列表变量可以用来定义一维或二维数组。下面我们就来创建一个列表变量。点击 Edit 标签页右侧的“Add List”按钮来创建变量



Name 定义变量名为：${student}

Value 填写列表变量的值：小明、小红、小华、小刚。

在测试用例中可以对这个一维数组进行遍历。

:FOR	${n}	in	@{student}	
	log	${n}		

执行结果：

```
Starting test: 测试项目.测试套件.test case3
20150304 14:06:01.138 :  INFO : 小明
20150304 14:06:01.141 :  INFO : 小红
20150304 14:06:01.142 :  INFO : 小华
20150304 14:06:01.144 :  INFO : 小刚
Ending test:   测试项目.测试套件.test case3
```

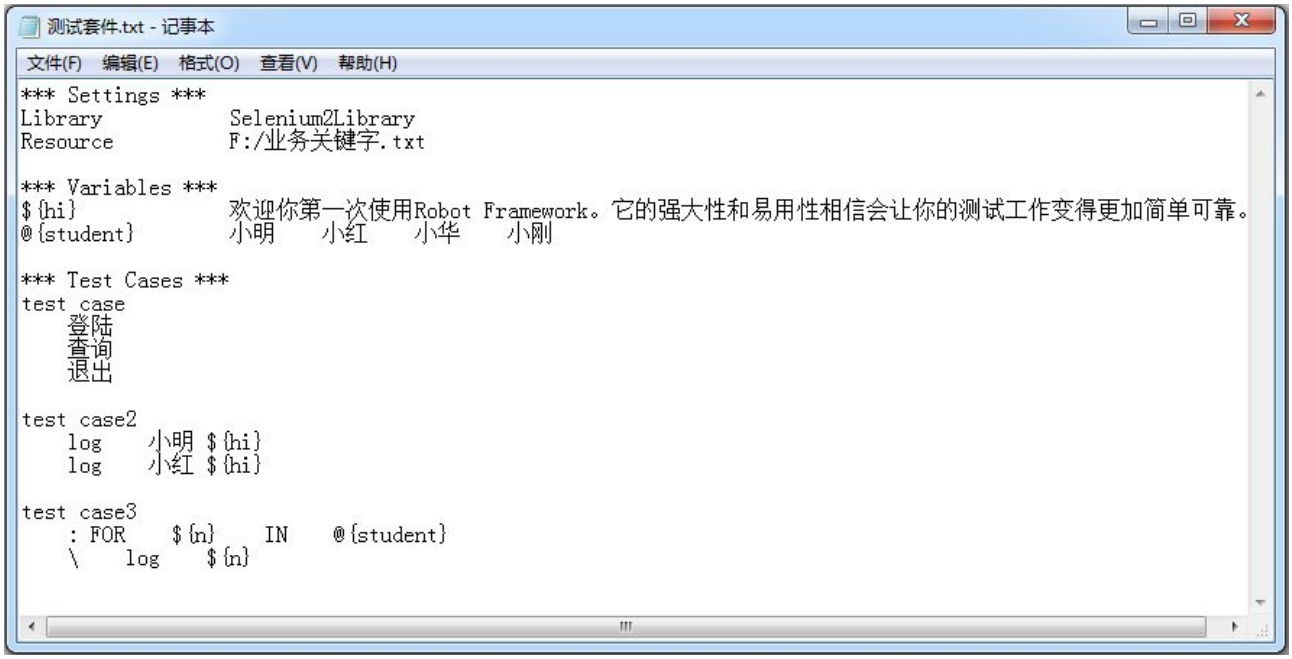
4.2 Text Edit 标签

我们在 Edit 标签页完成的工作，都可以在 Text Edit 标签页上完成。它们之间是对应关系，Edit 可视化的提供的按钮输入框，对于用户来说更容易知道我要怎么做；而在 Text Edit 中只是一个空当当的文本，我们跟本不知道如何下手。

好在我们已经在 Edit 中做了很多事情。切换到 Text Edit 将会看到这些信息的展示。

也许你会有点印象，在第一章第 3 节的例子就是这个样子的。

或者切换到“...\robot\测试项目”目录下，通过记事本打开“测试套件.txt”。



测试用例的本来面目也是这个样子的，只是在 RIDE 中对它进行的“美化”。

其实在这个在 Text Edit 下或第三方编辑器下编写 Robot Framework 测试的效率要远远高于 Edit 标签中的“填表格”式编写。读者可以在两种标签页之间切换来提高用例的开发效率。

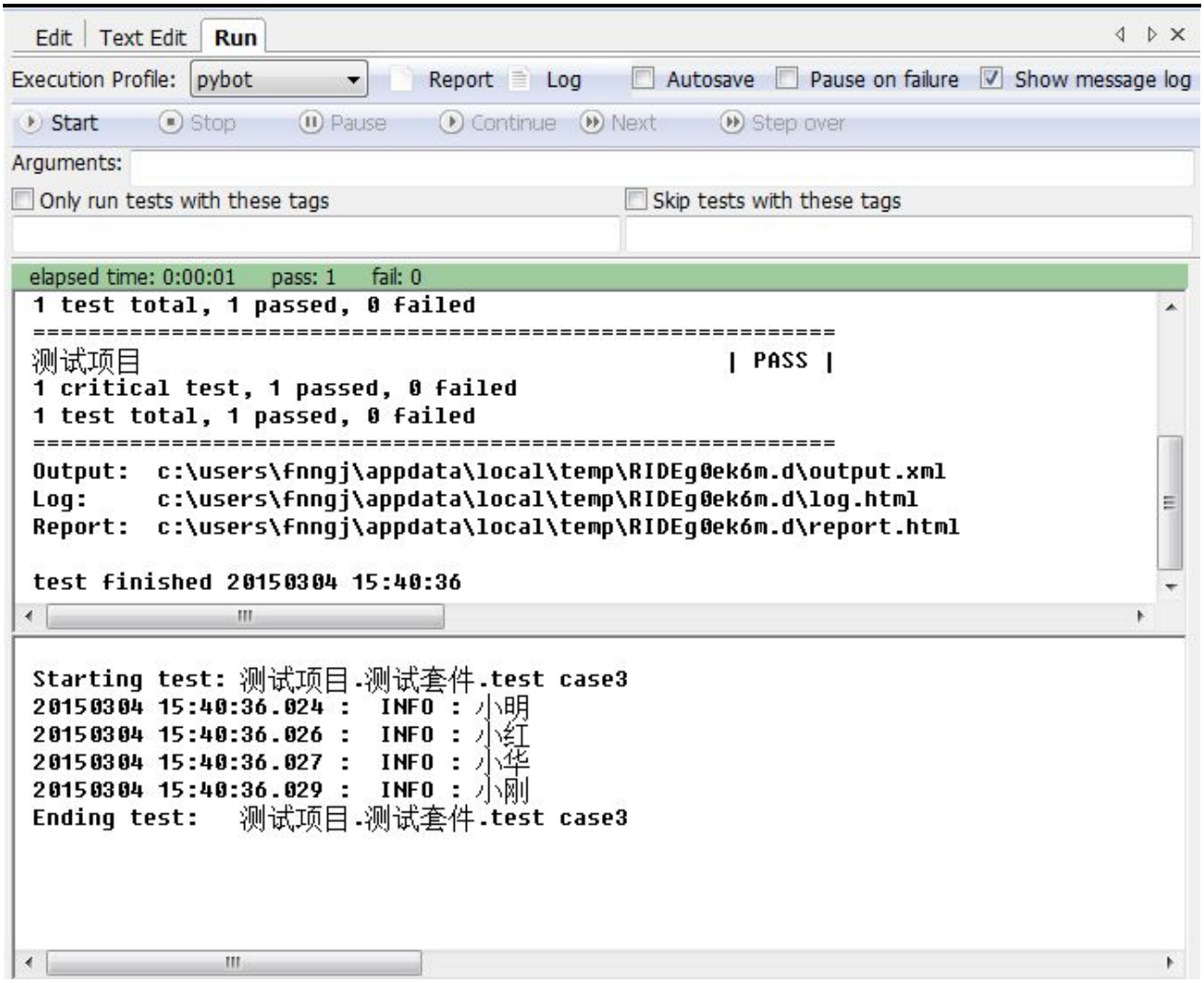
4.3 Run

我一直在想 Robot Framework 不要 RIDE 可不可以。对于编写测试用例来说，只要掌握 Robot Framework 的语法规则，随便在一个你顺手的编辑器下编写也没问题，甚至效率更高。为什么要填写那个该死的“表格”。

直到运行案例的时候我才意识到 RIDE 的好处。在 RIDE 中运行测试用例，就是勾选想要运行的用例，然后点击按钮即可。想想我们在做单元测试的时候可不会这么方便，调用 addTest()方法将一个个想要运行的测试方法添加到测试套件中，或者一行行的注释掉不添加到测试套件的测试用例的 addTest()方法，这是个极其痛苦的过程。

4.3.1 Run 标签

下面是 Run 标签的截图：



第一眼看上去，Run 标签提供了丰富的操作和日志。按照截图我们依次来说明 Run 标签上的按钮和输入框的作用：

1) Execution Profile：选择运行方式，里面有 pybot、jybot 和 custom script。其中我们默认是用 pybot 来运行案例，pybot 的运行 Python 编译器完成。jybot 需要安装 Jython 的支持。custom script 是选择自定义的脚本来运行。

2) Start 和 Stop：用例的运行和停止。

3) Report 和 Log：报告和日志，要运行之后才能点击。他们之间的区别：报告更多是结果上的展示，日志更多是过程的记录，在测试用例调试的过程中更多使用日志来查看执行错误。当只想知道测试用例的最终执行情况时用报告。

4) Autosave：自动保存，如果不勾选，在修改了用例之后如果没有保存的话，运行案例时会提示是否保存。勾选则在运行时自动保存了。

5) Arguments：pybot 的参数（或者 jybot 等），可以在这里输入 pybot 的命令完成相应的操作。

6) Only Run Tests with these Tags：只运行这些标记的测试案例。

7) Skip Tests with these Tags： 跳过这些标记的测试案例。

下面的两个区域，中间区域记录用例的执行过程，底部的区域输出用例的执行结果。

执行过程：

```
command: pybot.bat --argumentfile
c:\users\fnngj\appdata\local\temp\RIDEg0ek6m.d\argfile.txt --listener
C:\Python27\lib\site-packages\robotide\contrib\testrunner\TestRunnerAgent.py:
63009:False D:\robot\测试项目
==============================================================================
测试项目
==============================================================================
测试项目.测试套件
==============================================================================
test case                                                      | PASS |
------------------------------------------------------------------------------
test case2                                                     | PASS |
------------------------------------------------------------------------------
test case3                                                     | PASS |
------------------------------------------------------------------------------
测试项目.测试套件                                              | PASS |
3 critical tests, 3 passed, 0 failed
3 tests total, 3 passed, 0 failed
==============================================================================
测试项目                                                       | PASS |
3 critical tests, 3 passed, 0 failed
3 tests total, 3 passed, 0 failed
==============================================================================
Output:  c:\users\fnngj\appdata\local\temp\RIDEg0ek6m.d\output.xml
Log:     c:\users\fnngj\appdata\local\temp\RIDEg0ek6m.d\log.html
Report:  c:\users\fnngj\appdata\local\temp\RIDEg0ek6m.d\report.html

test finished 20150304 16:33:22
```

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150304 16:33:22.639 :  INFO : test1
Ending test:   测试项目.测试套件.test case

Starting test: 测试项目.测试套件.test case2
20150304 16:33:22.643 :  INFO : 小明 欢迎你第一次使用 Robot Framework。它的强大性和易用性相信会让你的测试工作变得更加简单可靠。
20150304 16:33:22.645 :  INFO : 小红 欢迎你第一次使用 Robot Framework。它的强大性和易用性相信会让你的测试工作变得更加简单可靠。
Ending test:   测试项目.测试套件.test case2
```

```
Starting test: 测试项目.测试套件.test case3
20150304 16:33:22.649 :  INFO : 小明
20150304 16:33:22.652 :  INFO : 小红
20150304 16:33:22.654 :  INFO : 小华
20150304 16:33:22.657 :  INFO : 小刚
Ending test:   测试项目.测试套件.test case3
```

4.3.2 运行与停止

在 Run 标签页提供了运行与停止的按钮，使用很简单。可是你知道到点击“运行”按钮的时候，Robot Framework 是怎么执行“测试套件.txt”文件的么？点击“停止”按钮的时候，Robot Framework 又做了什么操作来终止用例的执行的？带着这样的疑问，我们来简单的读一下 RIDE 的 run 代码。

首先打开 C:\Python27\Lib\site-packages\robotide\run 目录下的 process.py 文件。

process.py

```
import os
import time
import tempfile
import subprocess


class Process(object):
……
    def start(self):
        self._out_fd, self._out_path = \
                    tempfile.mkstemp(prefix='rfproc_', suffix='.txt',
                                     text=True)
        self._out_file = open(self._out_path)
        if not self._command:
            self._error = 'The command is missing from this run configuration.'
            return
        try:
            self._process = subprocess.Popen(self._command, stdout=self._out_fd,
                                             stderr=subprocess.STDOUT)
        except OSError, err:
            self._error = str(err)

……


    def stop(self):
        try:
```

```
            self._process.kill()
        except AttributeError:
            raise AttributeError('Stopping process is possible only with '
                                 'Python 2.6 or newer')
```

……

```
    def get_output(self, wait_until_finished=False):
        """Returns the output produced by the process.

        If ``wait_until_finished`` is True, blocks until the process is
        finished and returns all output. Otherwise the currently available
        output is returned immediately.

        Currently available output depends on buffering and might not include
        everything that has been written by the process.
        """
        if self._error:
            self._close_outputs()
            return self._error
        if wait_until_finished:
            self._process.wait()
        output = self._out_file.read()
        if self.is_finished():
            self._close_outputs()
        return output

    def _close_outputs(self):
        self._out_file.close()
        os.close(self._out_fd)
        self._remove_tempfile()
```

以上为 process.py 文件的部分代码。看有标蓝的代码即可。

首先看 start()方法，通过 tempfile 模块的 mkstemp()方法找到“txt”文件，也就是“测试套件.txt”文这类件。接着通过 open()方法打开。

在 get_output()方法中通过 read()方法来读取“txt”文件。最后把读取的文件的赋值给变量 output 并返回。

在_close_outputs()方法中通过 close()关闭打开的“txt”文件。

停止测试用例的执行非常单间，由 stop()方法实现，通过调用 kill()将用例的执行进程杀死。

代码读到这里只是开始，get_output()方法读到的文件返回给谁了呢？或者谁会调用 get_output()方法呢？继续打开 C:\Python27\Lib\site-packages\robotide\run 目录下的 ui.py 文件

ui.py

```
……
from robotide.run.process import Process

……
class Runner(wx.EvtHandler):

    def __init__(self, config, notebook):
        wx.EvtHandler.__init__(self)
        self.Bind(wx.EVT_TIMER, self.OnTimer)
        self.name = config.name
        self._timer = wx.Timer(self)
        self._config = config
        self._window = self._get_output_window(notebook)

    def _get_output_window(self, notebook):
        return _OutputWindow(notebook, self)

    def run(self):
        self._process = Process(self._config.command)
        self._process.start()
        self._timer.Start(500)

    def OnTimer(self, event=None):
        finished = self._process.is_finished()
        self._window.update_output(self._process.get_output(), finished)
        if finished:
            self._timer.Stop()

    def stop(self):
        try:
            self._process.stop()
        except Exception, err:
            wx.MessageBox(str(err), style=wx.ICON_ERROR)
……
```

ui.py 文件调用 process.py 文件的方法来运行测试用例。如果读者精通于 Python 语言的话可以顺着这条线继续读下去，看看哪个方法会调用 Runer 类。因为本文档的重点的不是分析 Robot Framework 代码，所以不在再继续。但这里想传达的思路是知其然，一定要知其所以然；用工具而不要受制于工具。

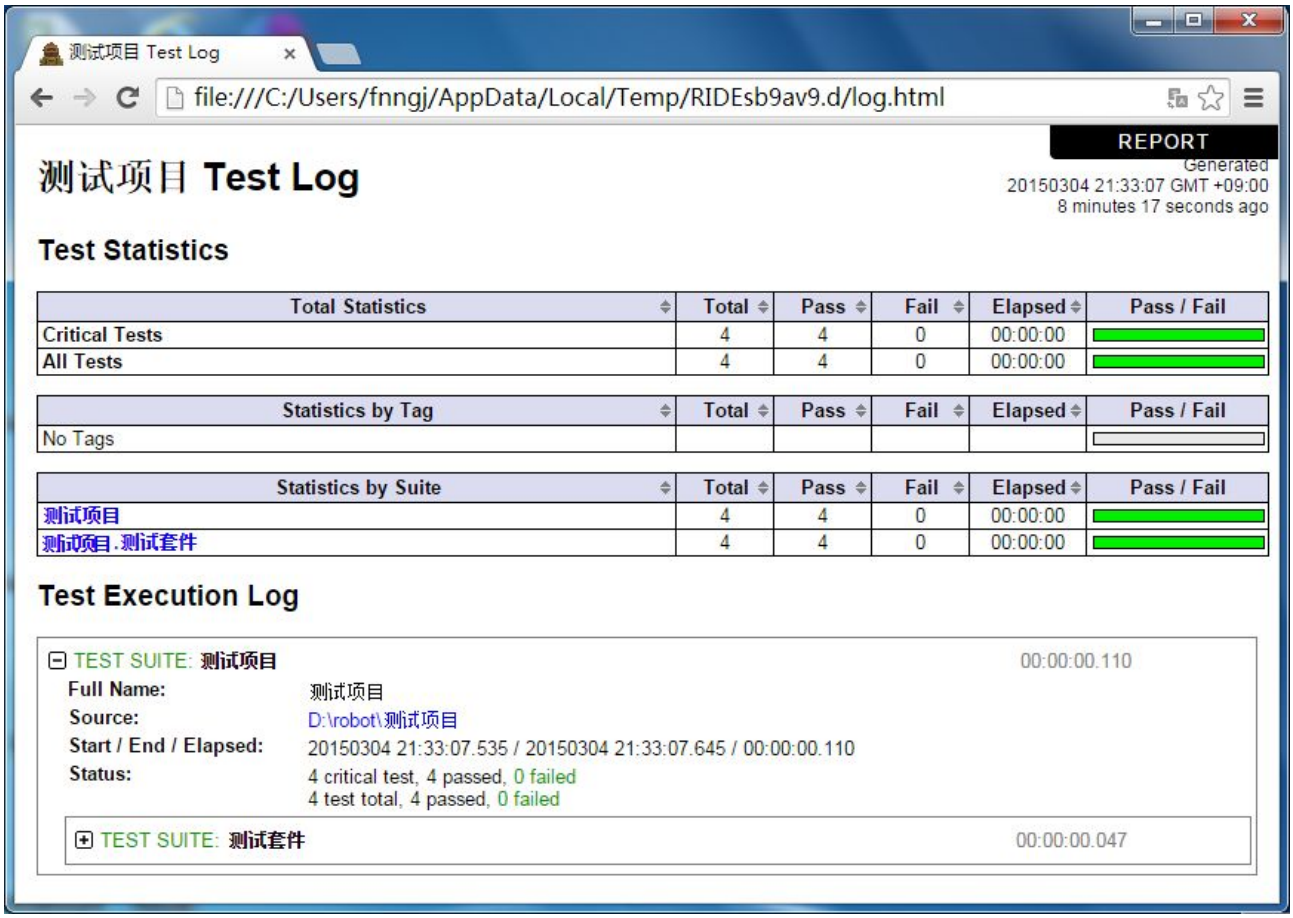
4.3.3 报告与日志

当用例运行结束，Robot Framework 生成三个文件：output.xml、log.html 和 report.html。

output.xml 记录的测试结果是 xml 文件，这个文件不够直观。根据特定的需要可以编写脚本读取 xml 文件并生成特定的测试报告。

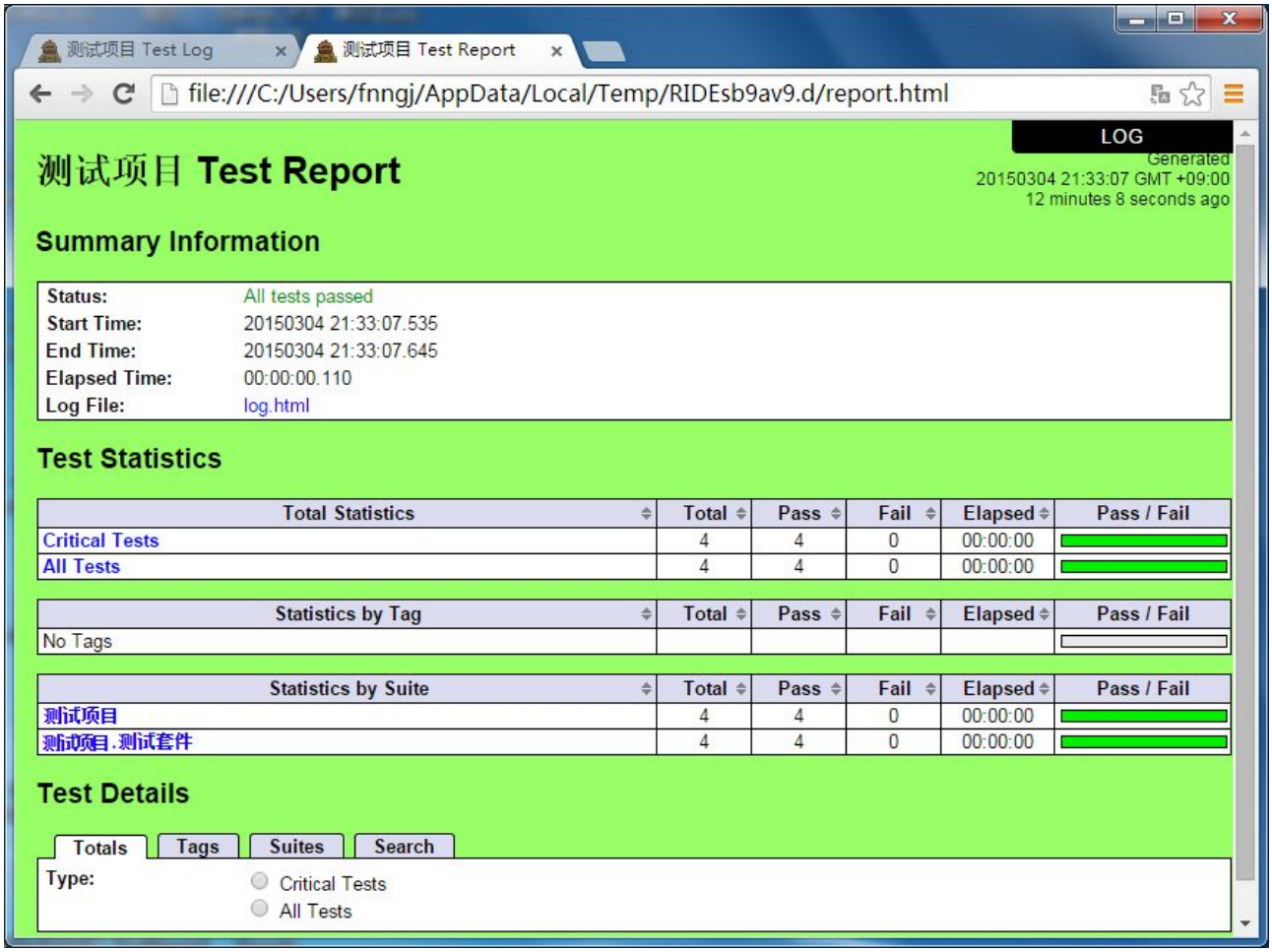
相比较而言 log.html 和 report.html 报告要直观得多，因为是 html 格式的嘛。

查看 log.html 文件，点击 Run 标签而上的“Log”按钮，通过默认浏览器打开。



在 log.html 文件中可以查看用例执行的每一步，适合跟踪定义问题。

查看 report.html，点击 Run 标签而上的“Report”按钮，通过默认浏览器打开。



report.html 用于最终结果的展示，适合了解测试用例的执行情况：测试了哪些模块，用例数、失败率等。

4.3.4 筛选执行用例

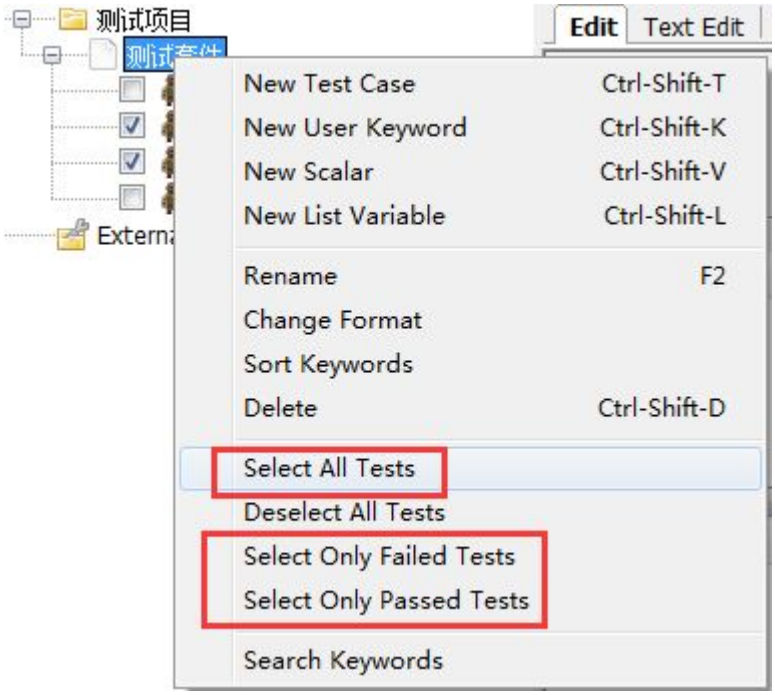
这一节来探讨一下，几中方式可以筛选要运行的运用例。

第一种：勾选用例

在要执行的用例前面打勾。

这种方法最简单和直观，要运行哪条用例就勾选哪一条。如果全部不勾选，点击“运行”按钮会运行所有用例。

也可以在“测试套件”上右键选择：



Select All Test：选择当前套件的所有用例。

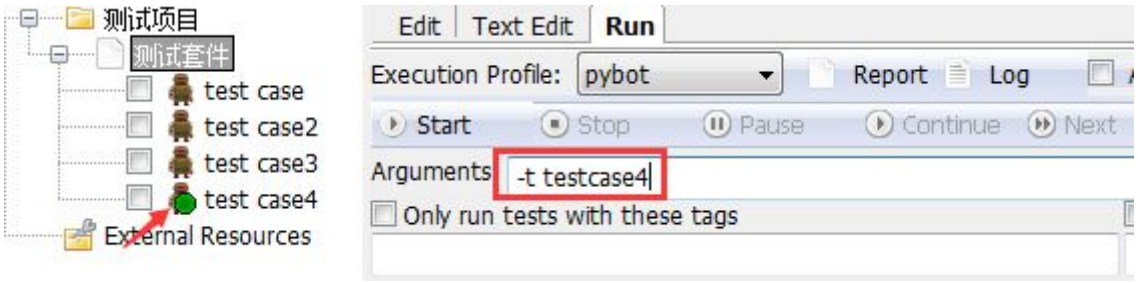
Select Only Failed Test：选择当前套件下运行失败的用例。

Select Only Passed Test：选择当前套件下运行成功的用例。

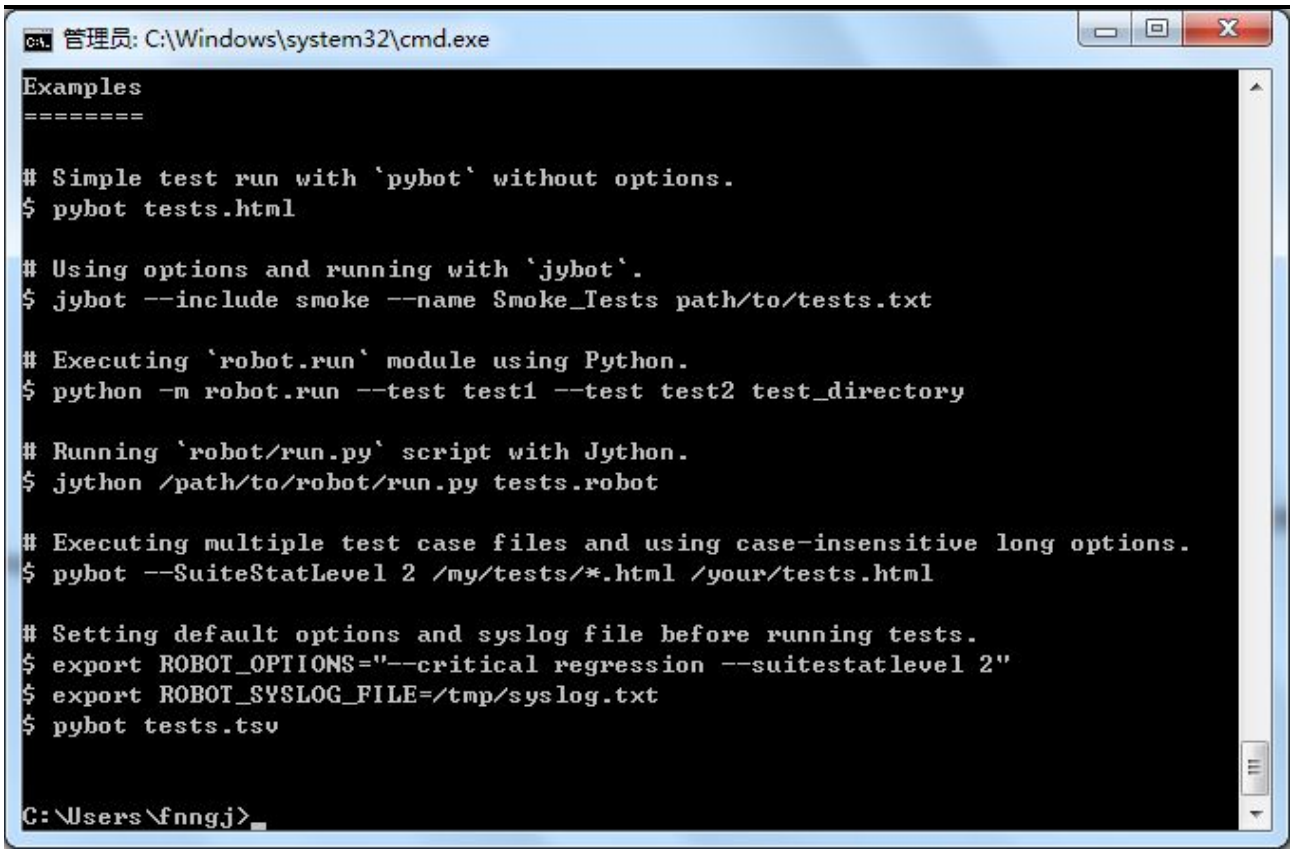
对于一个测试套件下有几十上百个用例来说，这几选项将非常有用。

第二种：用命令

这就用到 Run 标签中的 Arguments 功能。



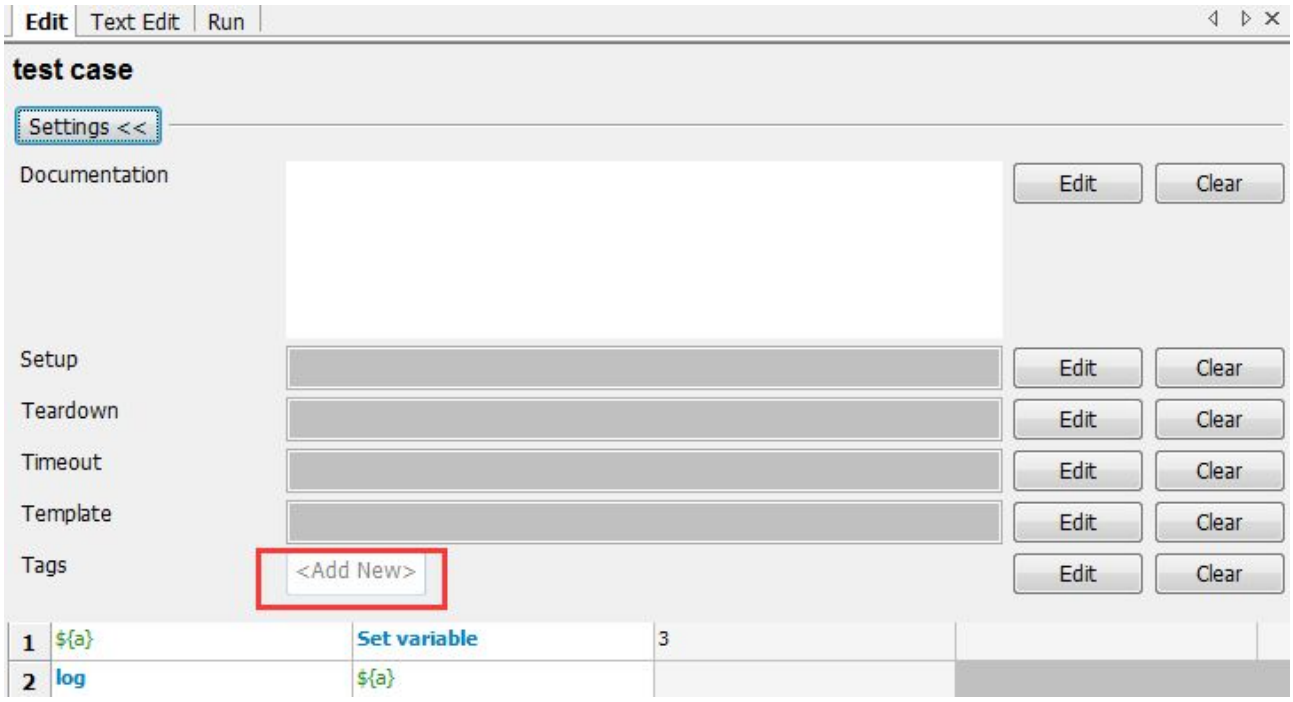
在 Arguments 的输入框内输入“-t testcase4”。点击“Start”按钮，只执行了 test case4 这一条用例。Arguments 能做的事情可不止于此。想了解更多多命令。可以在 cmd.exe 下执行“pybot.bat --help”。



第三种：筛选标记

这种方式就非常有意思的，对于不同的人来说会有一些标记，比如某富二代的标记就是“任性”。对于用例来说也可以打上标记。比如“重要”、“一般”、“基础”等。

点击某个用例，你会看到“Setting>>”的按钮，点击按钮展开：

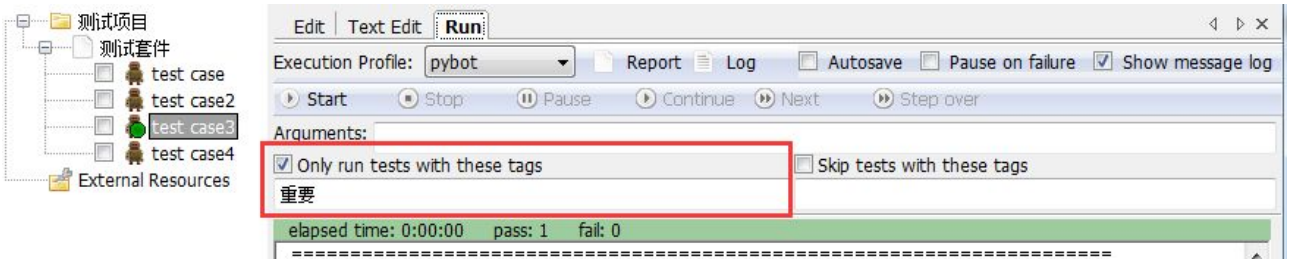


在最下面将会看到“Tags”的选项，在“<Add New>”的输入框内输入“重要”。这条用例就打上了“重要”的标记。

现在切换到 Run 标签，我要运行带“重要”标记的用例了，如何去做了？这就要用到：

Only Run Tests with these Tags：只运行这些标记的测试案例。

Skip Tests with these Tags： 跳过这些标记的测试案例。



因为“test case3”被打上了“重要”的标记，所以它被执行了。

对于一个用例来说，我们可以为它添加多个标记。勾选“Skip Tests with these Tags”选项可以跳过某些标记的用例。

4.4 Settings

不管是测试套件还是测试用例都会有一个“Settings>>”的按钮，因为它默认是被折叠起来的，所以，一般不太容易发现它，更不知道点击它之后是可以展开的。

4.4.1 测试用例的 Settings

点击测试用例上的“Settings>>”按钮，会看到下面的选项。

Documentation：用于描述用例的一个小文本，它可以把 URL 地址转换为可点击的链接。

Setup 和 Teardown：如果你了解 unittest 单元测试框架的话，一定对这两个单词不陌生。

setUp 用于设置初始化工作，在每一个测试用例前先被执行

tearDown 方法在每个测试方法执行后调用，这个方法用于完成测试用例执行后的清理工作，如执行“close browser”关闭浏览器等。

TimeOut：用于设置用例的超时时间。如“1 min 10s”、“ 2 hours”等。

Template：指定模板使用的关键字。

Tags ：用于给用例添加标记。在上一小节中有用到这个功能。

4.4.2 测试套件的 Settings

同样点击测试套的“Setting>>”按钮打开套件设置：

这里的 Setup 和 Teardown 分测试套件的和测试用例的。“Suite Setup”和“Suite Teardown”用于当前套件的开始和结束所要做的事情。“Test Setup”和“Test Teardown”会作用于套件下每一个测试用例开始和结束所要做的事情。

Force Tags：表示当前测试套件下测试用例强制的标记。

Default Tags：表示当前测试套件下测试用例默认的标记。

4.5 用户关键字

在 Robot Framework 中关键字的创建分两种：系统关键字和用户关键字。系统关键字需要通过脚本开发相应的类和方法，这个我们将在后面的章节介绍。用户关键字的创建就要简单得多，它主要利用现有的系统关键字，根据不同的业务，把多个重复的步骤集合在一起组成用户关键字。

比如，我们有一个循环：

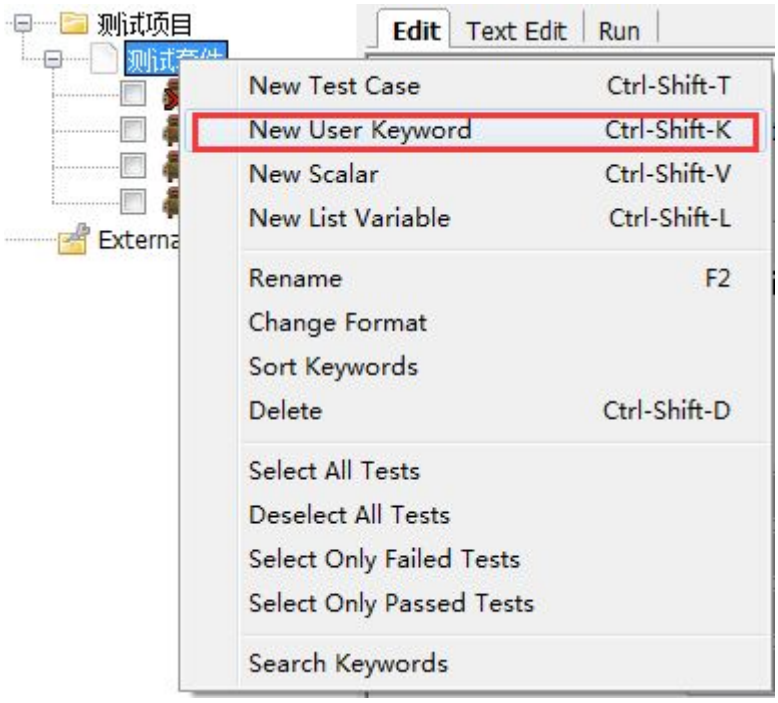
:FOR	${i}	in range	10
	log	${i}	

我们经常会用到这个循环，只是每次循环的次数不是一样。有时候需要循环 5 次，有时候需要循环 8 次。对这样的需求，我们就可以将这个循环封装成一个关键字。

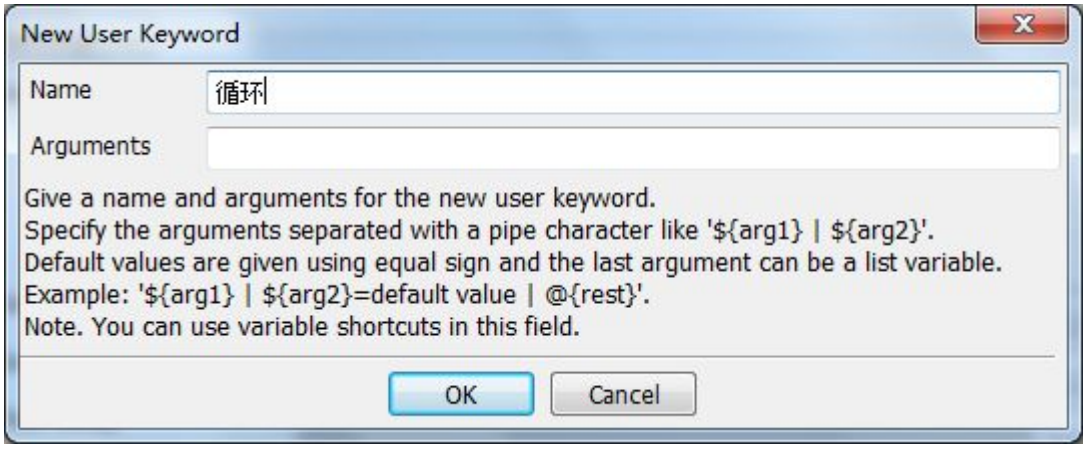
右键点击“测试项目”或“测试套件”都可以看到“New User Keyword”的选项。我们可以根据自己的需求选择在项目或套件下创建用户关键字。

4.5.1 创建用户关键字

在测试套件上右键选择“New User Keyword”选项。



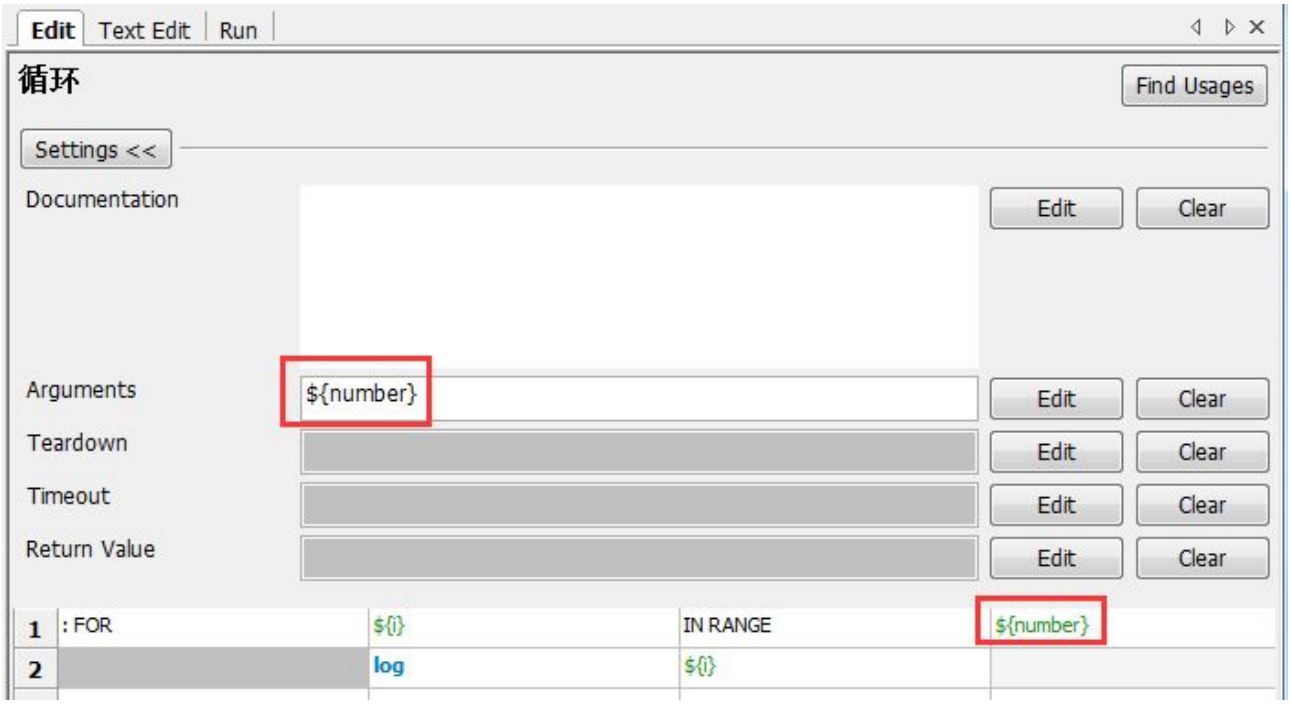
在弹出的菜单中输入用户关键字的名称。



查看左侧项目列表，用户关键字已经创建完成。

注意这个时候用户关键字和测试用例平级，所以在测试用例中可以直接使用。

下面编写用户关键字。（用户关键字的 Edit 标签与测试用例的 Edit 标签很像。）



点击“setting>>”按钮，Arguments 参数为：${number} ，这就像定函数的输入参。可以设置多个变量，之间用“|”分隔。

添加循环的用例，循环的次数为：${number}

:FOR	${i}	in range	${number}
	log	${i}	

接着在用例中使用“循环”用户关键字。

执行结果：

```
Starting test: 测试项目.测试套件.test case4
20150305 10:56:17.547 :  INFO : 0
20150305 10:56:17.563 :  INFO : 1
20150305 10:56:17.563 :  INFO : 2
20150305 10:56:17.563 :  INFO : 3
20150305 10:56:17.563 :  INFO : 4
Ending test:   测试项目.测试套件.test case4
```

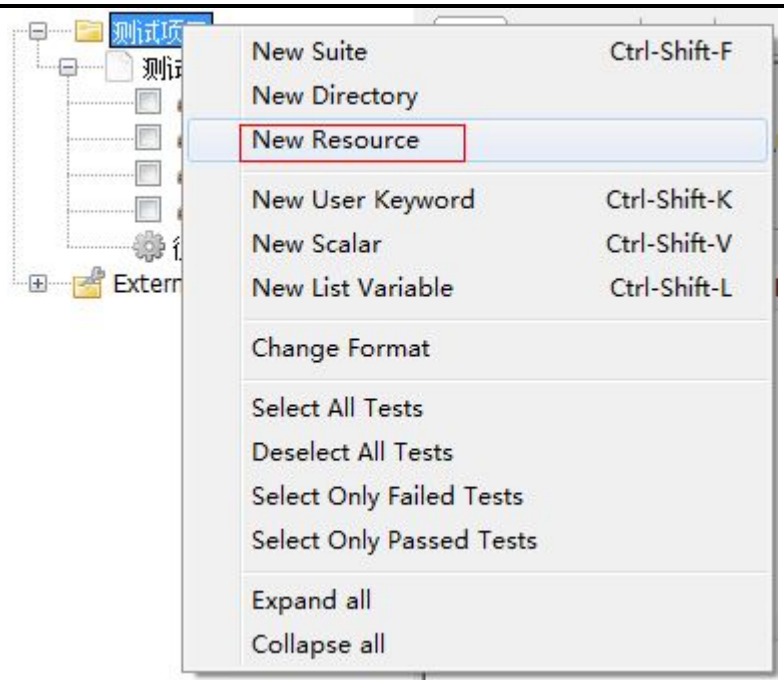
将循环的次数改为“8”，运行用例将循环 8 次。

4.5.2 创建资源

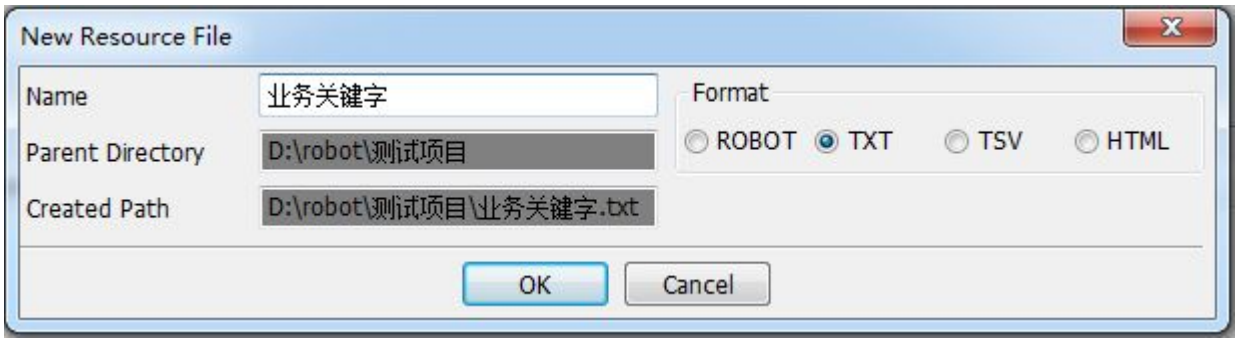
在实际的项目测试过程中，我们往往需要创建许多这样的具有通用性的用户关键字。不希望这些关键字依附于某个测试套件，甚至是某个项目。那么我们可以创建资源文件用于存放这些关键字。

1、创建资源

右键“测试项目”选择“New Resource”创建资源。

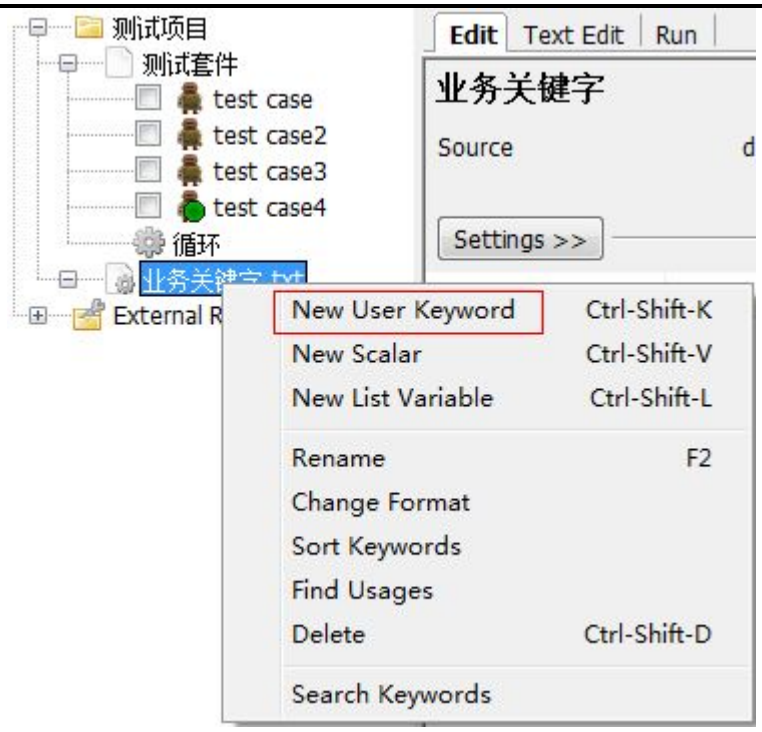


输入资源名称：



2、创建关键字

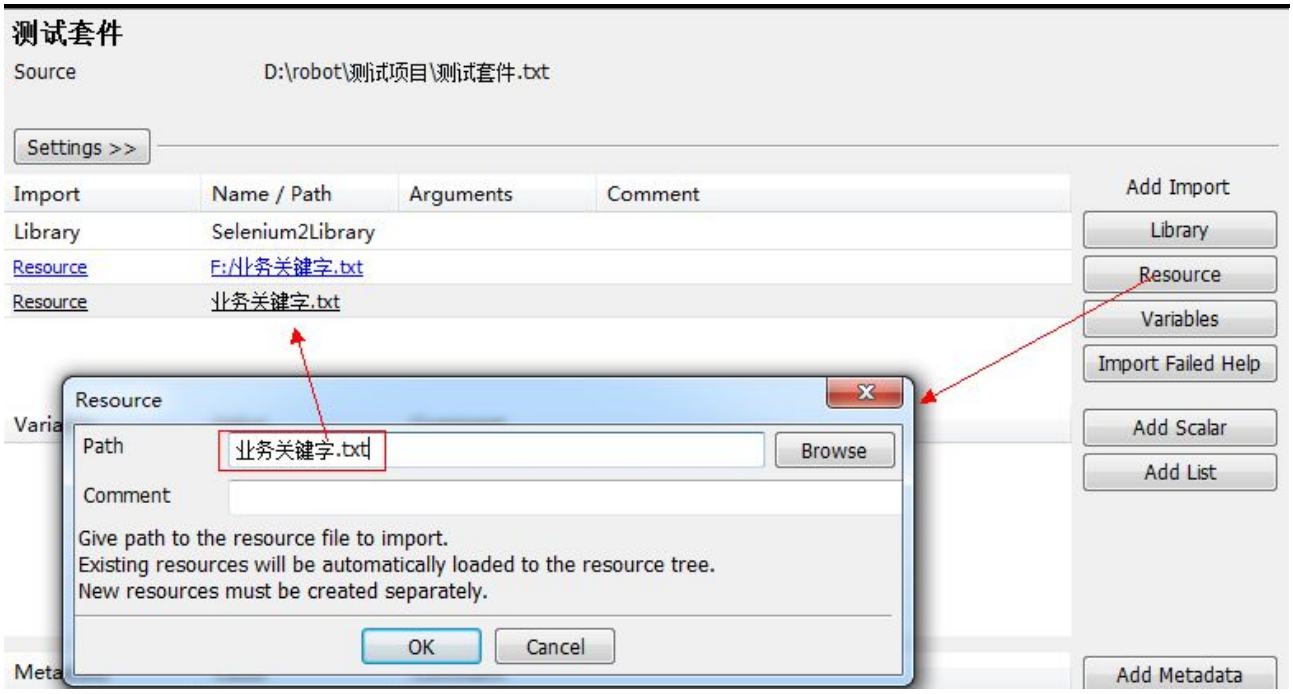
右键“业务关键字”选择“New User Keyword” 来创建用户关键字。



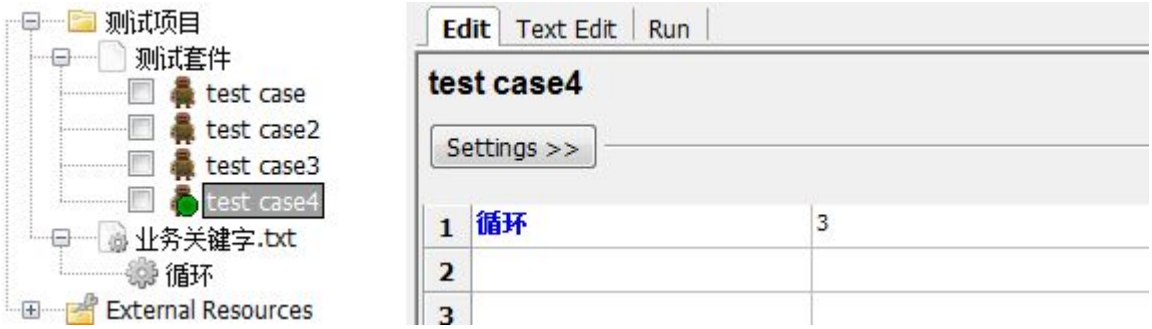
创建完成的项目结构如下：

3、导入资源

因为“业务关键字.txt”和“测试套件”属于并列关系。测试套件要想使用业务关键字下的“循环”关键字，需要导入资源。



现在就可以在测试用例中使用“循环”关键字了。



小结：

通过这一章的学习，RIDE 已经基本掌握了使用。后面的学习将以 Robot Framework 和 RIDE 为基础，介绍扩展库的使用，比如，接下来要学习的 Selenium2Library。

第 5 章 Selenium2Library 库

Selenium 是非常流行的开源 web 自动化测试工具，对于大多使用 Robot Framework 框架的人都会有使用 Selenium2Library 库来进行 web 自动化测试工具。所以，这一章就来学习 Selenium2Library 库。

5.1 Selenium

5.1.1 Selenium 介绍

Selenium 自动化测试工具，它主要是用于 Web 应用程序的自动化测试，但并不只局限于此，同时支持所有基于 web 的管理任务自动化。

Selenium 的特点：

- 开源，免费
- 多浏览器支持：FireFox、Chrome、IE、Opera
- 多平台支持：linux 、windows、MAC
- 多语言支持：java、Python、Ruby、php、C#、JavaScript
- 对 web 页面有良好的支持
- 简单（API 简单）、灵活（用开发语言驱动）
- 支持分布式测试用例执行

Selenium 是支持多种开发语言的，对于不同的语言来说都有其对应的库。

对 Robot Framework 框架的 Selenium 库有两个：SeleniumLibrary 和 Selenium2Library。SeleniumLibrary 是基于 Selenium1.0 开发的，Selenium2Library 是基于 Selenium2.0 开发的。如果没有历史遗留问题，我们直接使用 Selenium2Library。

5.1.2 安装 Selenium2Library

下载地址:https://pypi.python.org/pypi/robotframework-selenium2library/1.5.0

可以通过下载 exe 程序进行安装，Robot framework-selenium2library 分别提供了，win-amd64.exe 和 win32.exe 两个 Windows 版本，你可以根据自己的环境下载相应的版本，双击进行安装。

如果像安装普通的 Python 程序，可以下载 tar.gz 文件，解压并运行 setup.py 文件进行安装。

cmd.exe

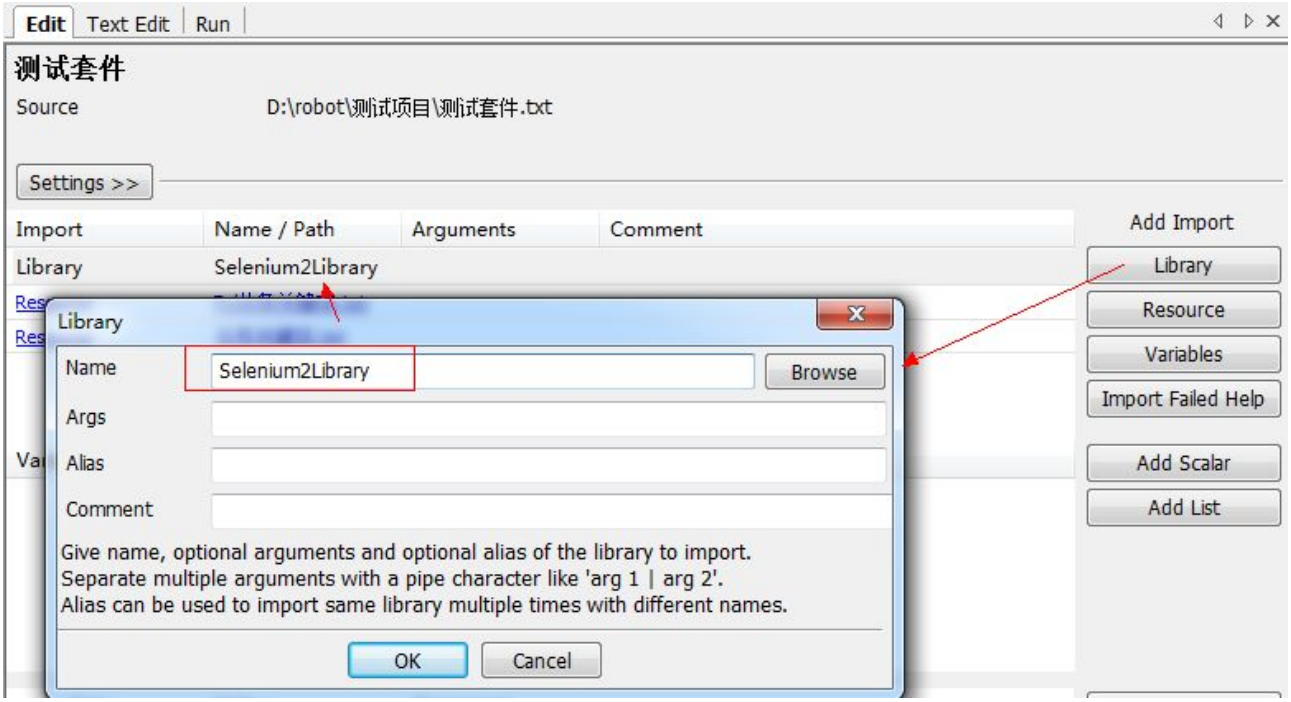
```
C:\robot\Robot framework-selenium2library-1.5.0>python setup.py install
```

因为在上一小节中我们已经安装了 pip，所以通过 pip 命令安装更为方便和快捷：

cmd.exe

```
C:\Python27\Lib\site-packages>pip install robotframework-selenium2library
```

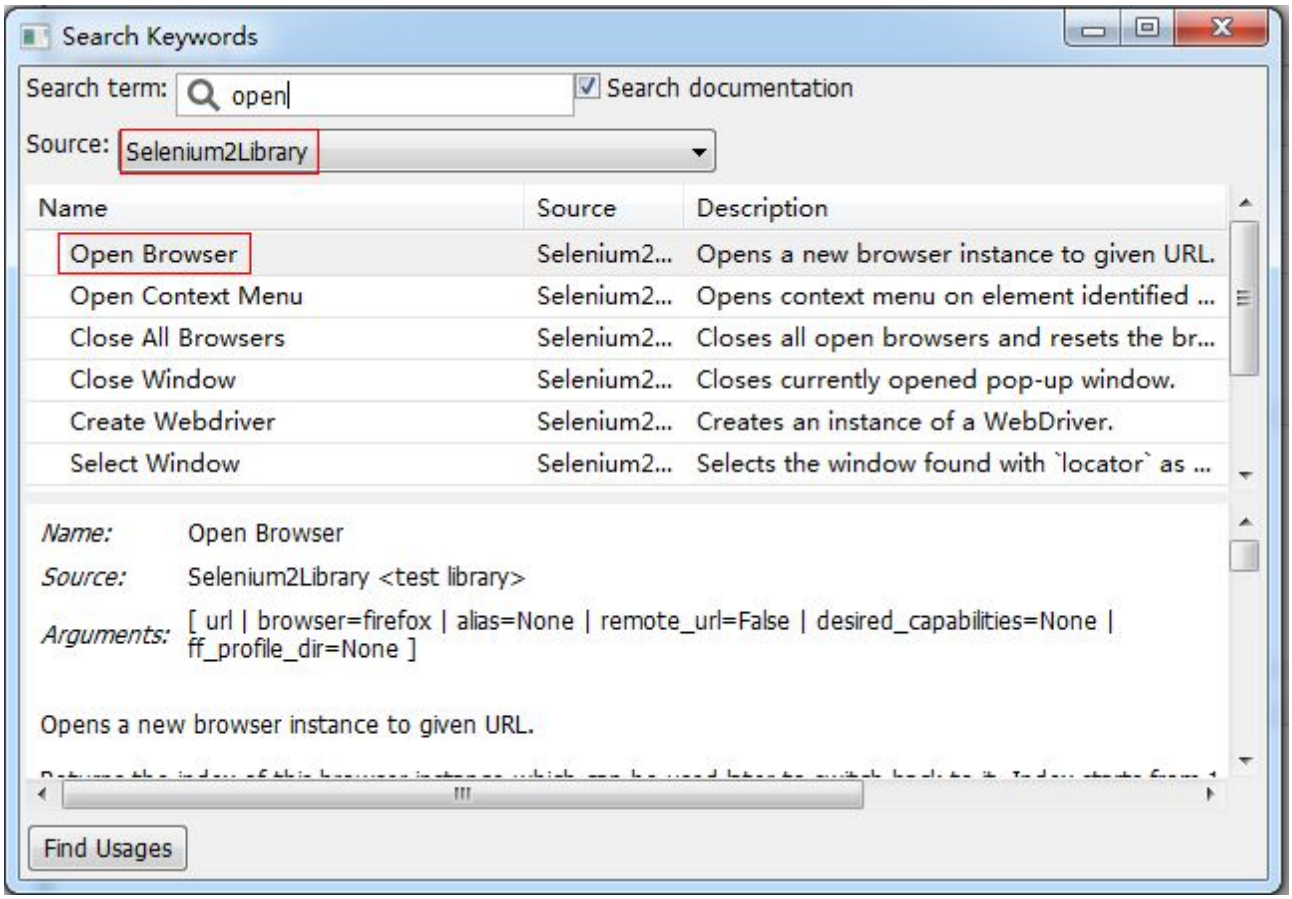
在上一章中已经知道了如何添加库，现在我们将“Selenium2Library”库添加到相应的测试套件中。



添加完成，黑色示添加的库正常，红色表示库不存。如果为红色，请检查 C:\Python27\Lib\site-packages 目录下是否有 Selenium2Library 目录

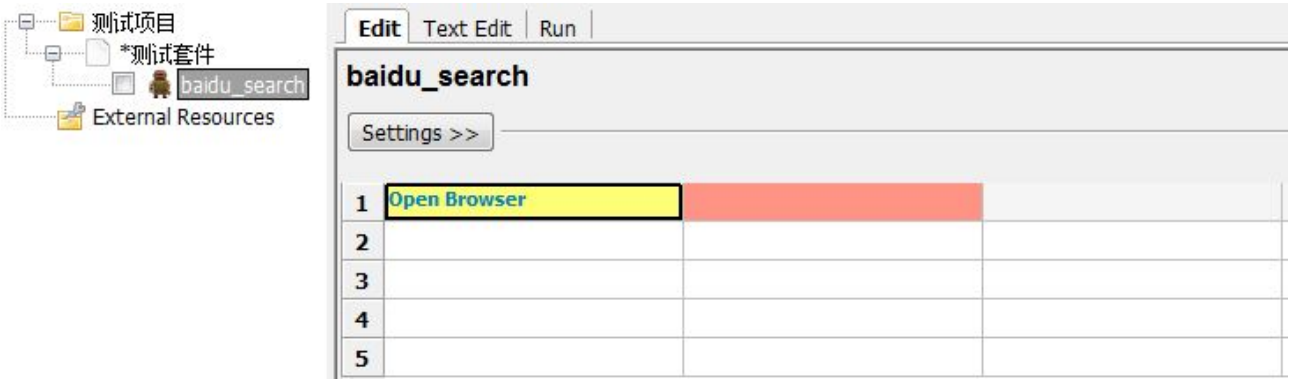
5.1.3 第一个例子

我们已经有了学习 Robot Framework 的经验，通过按 F5 快捷键来查询库所提供的关键字。



如上图，自动化脚本从打开浏览器开如，我想打开一个浏览器，自然想到的是以“open”为关键字进行搜索，结果找到了一个“Open Browser”的关键字，点击这个关键字，显示它的用法和说明。

根据说明，我们来尝试创建这个打开浏览器的操作吧：



“Open Browser”变蓝了，说明它是一个合法的关键字，后面有一个方框是红色的，表示这个参数不能缺省的。通过说明信息中，我发现它需要一个 url 地址是必填的，当然还需要指定 browser （默认不填为 friefox）

更多关键的使用后面会讲；这里跟新着我们写一个 web 自动化测试用例（百度搜索用例）。

open browser	http://www.baidu.com	
input text	id=kw	robot framework 学习
click button	id=su	
close browser		

执行结果：

```
Starting test: 测试项目.测试套件.baidu_search
20150305 18:07:21.169 :  INFO : Opening browser 'chrome' to base url
'http://www.baidu.com'
20150305 18:07:49.051 :  INFO : Typing text 'robot framework 学习' into text field
'id=kw'
20150305 18:07:50.406 :  INFO : Clicking button 'id=su'.
Ending test:   测试项目.测试套件.baidu_search
```

5.2 元素定位

对于 Web 自动化测试来说，就是操作页面上的各种元素，在操作元素之间需要先找到元素，换句话说就是定位元素。

Selenium2Library 提供了非常丰富的定位器：

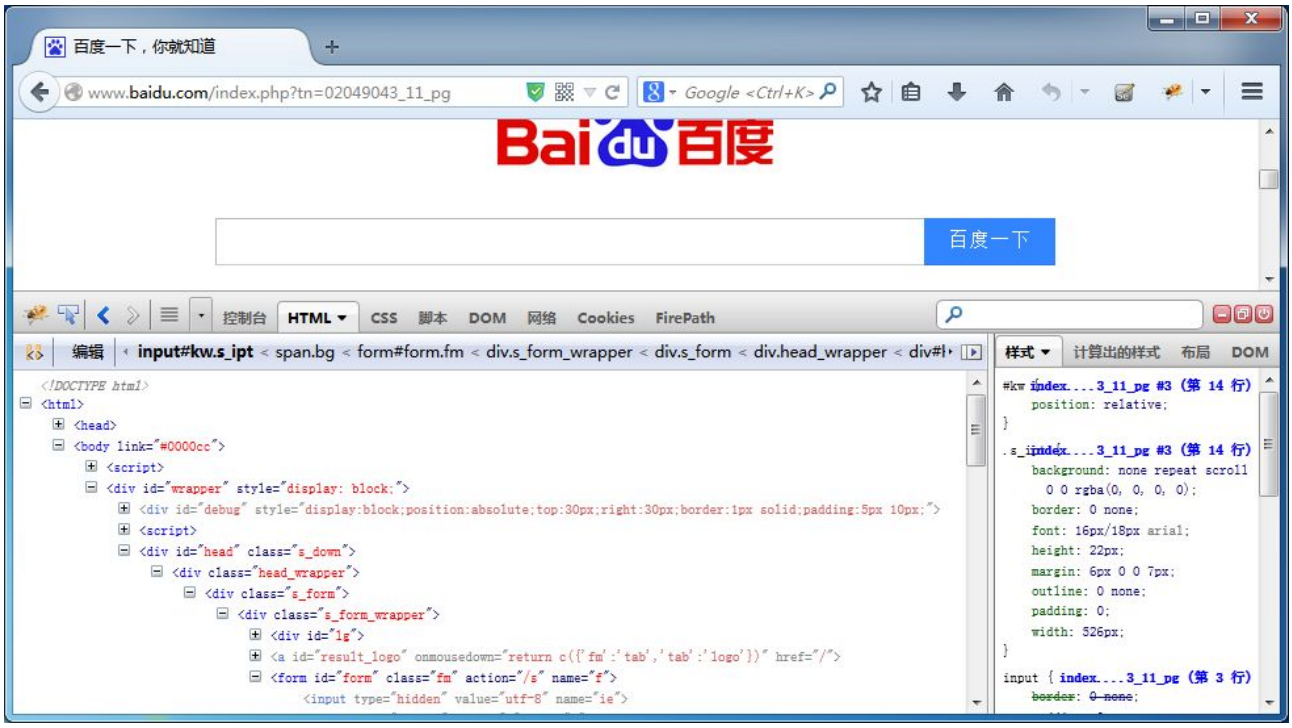
Strategy	Example	Description
identifier	Click Element | identifier=my_element	Matches by @id or @name attribute
id	Click Element | id=my_element	Matches by @id attribute
name	Click Element | name=my_element	Matches by @name attribute
xpath	Click Element | xpath=//div[@id='my_element']	Matches with arbitrary XPath expression
dom	Click Element | dom=document.images[56]	Matches with arbitrary DOM express
link	Click Element | link=My Link	Matches anchor elements by their link text
partial link	Click Element | partial link=y Lin	Matches anchor elements by their partial link text
css	Click Element | css=div.my_class	Matches by CSS selector
jquery	Click Element | jquery=div.my_class	Matches by jQuery/sizzle selector
sizzle	Click Element | sizzle=div.my_class	Matches by jQuery/sizzle selector
tag	Click Element | tag=div	Matches by HTML tag name
default*	Click Link | default=page?a=b	Matches key attributes with value after first '='

虽提供了这么多种定位方式，并不是要求我们每一种都要学会。在这里我只介绍 4 种定位方式，id、name、xpath 和 css。介绍 id 和 name，是因为这两种定位方式非常简单且实用，介绍 xpath 和 css，是因为这两种定位方式足够强大，可以满足几乎所有定位需求。

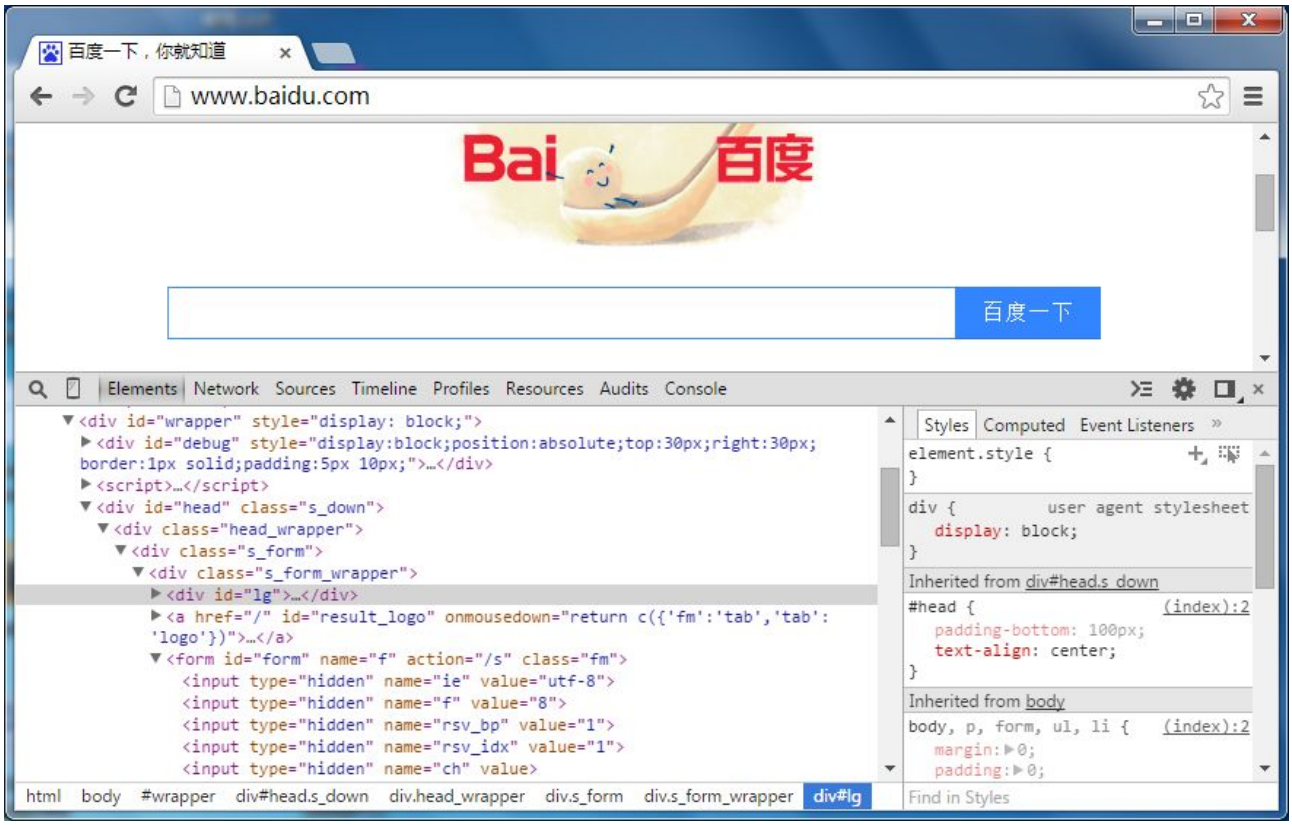
5.2.1 前端工具

在学习定位之前，有必要先介绍一下浏览器的前端具。

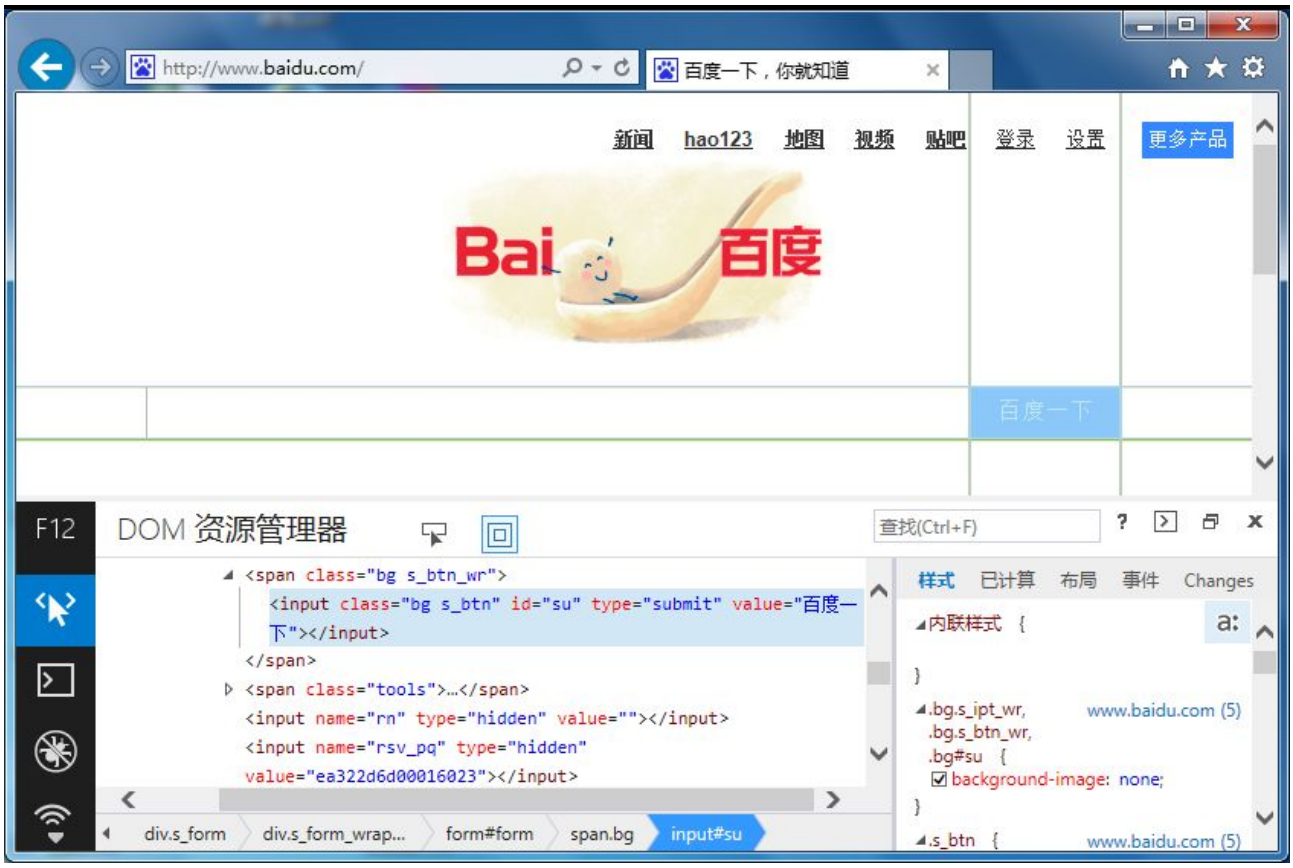
firefox 浏览器可以通过 firebug 工具查查看页面元素。



chrome 浏览器可以通过 F12 快捷打开自带的前端工具查看页面元素：



IE 浏览器也可以通过 F12 快捷打开自带的前端工具查看页面元素：



baidu.html

```
<html>
   <head>
   <body>
       <script>
       <div id="wrapper" style="display: block;">
          <div id="debug" style="display:block;position:..">
          <script>
          <div id="head" class="s_down">
             <div class="head_wrapper">
                <div class="s_form">
                   <div class="s_form_wrapper">
                      <div id="lg">
                      <a id="result_logo" onmousedown="return .." href="/">
                      <form id="form" class="fm" action="/s" name="f">
                         <input type="hidden" value="utf-8" name="ie">
                         <input type="hidden" value="8" name="f">
                         <input type="hidden" value="1" name="rsv_bp">
                         <input type="hidden" value="1" name="rsv_idx">
                         <input type="hidden" value="" name="ch">
                         <input type="hidden" value="02.." name="tn">
                         <input type="hidden" value="" name="bar">
```

```
                        <span class="bg s_ipt_wr">
                            <input id="kw" class="s_ipt" autocomplete="off"
                                   maxlength="100" value="" name="wd">
                        </span>
                        <span class="bg s_btn_wr">
                            <input id="su" class="bg s_btn" type="submit"
                                value="百度一下">
                        </span>
....
    </body>
</html>hello
```

注意这段代码并非百度首页的页面源代码，而是通过前端工具查看所得到页面代码与结构。那么这样的 HTML 结构有如下特征。

5.2.2 id 和 name 定位

假如把一个元素看作一个人的话，id 和 name 可以看作一个人的身份证号和姓名。当然，这些属性值是否唯一要看前端工程师如何设计了。

根据上面的例子，百度输入框可以取 id 或 name 进行定位。（前提是 id 和 name 的值在当页面上唯一）

id = kw1

name = wd

在 Robot framework 中就是这样写的：

input text	id=kw	robot framework 学习
input text	name=wd	robot framework 学习

Input text 用于输入框的关键字，“robot framework 学习”是要给输入框输入的内容。

百度按钮只 id 数据可以利用：

id=su

click button	id=su	

Click Button 是按钮点击的关键字。

5.2.3 xpath 定位

XPath 是一种在 XML 文档中定位元素的语言。因为 HTML 可以看做 XML 的一种实现，所以 selenium 用户可是使用这种强大语言在 web 应用中定位元素。

假如，一个人没身份证号没名字怎么找呢？想想你是怎么找朋友吃饭的，他手机不通，电话不回呢？直接上他家去呗，那你一定有他家住址，xx 市 xx 区 xx 路 xx 号。xpath 就可以通过这种层级关系找到元素。

1、xpath 的绝对路径：

xpath = /html/body/div[1]/div[4]/div[2]/div/form/span[1]/input

我们可以从最外层开始找，html 下面的 body 下面的 div 下面的第 4 个 div 下面的....input 标签。通过一级一级的锁定就找到了想要的元素。

2、xpath 的相对路径：

绝对路径的用法往往是在我们迫不得已的时候才用的。大多时候用相对路径更简便。

2.1、元素本身：

Xpath 同样可以利用元素自身的属性：

Xpath = //*[@id='kw1']

//表示某个层级下，*表示某个标签名。@id=kw1 表示这个元素有个 id 等于 kw1 。

当然，一般也可以制定标签名：

Xpath = //input[@id='kw1']

元素本身，可以利用的属性就不只局限为于 id 和 name ，如：

Xpath = //input[@type='text']

Xpath = //input[@autocomplete='off']

但要保证这些元素可以唯一的识别一个元素。

2.2、找上级：

当我们要找的一个人是个刚出生的婴儿，还没起名子也没有入户口（身份证号），但是你会永远跟在你父亲的身边，你的父亲是有唯一的名字和身份证号的，这样我们可以先找到你父亲，自然就找到你的。

元素的上级属性为：

```
<form id="form1" class="fm" action="/s" name="f1">
<span class="bg s_ipt_wr">
    <input id="kw1" class="s_ipt" type="text" maxlength="100" name="wd" utocomplete="off">
```

找爸爸：

xpath = //span[@class='bg s_ipt_w']/input

如果爸爸没有唯一的属性，可以找爷爷：

xpath = //form[@id='form1']/span/input

这样一级一级找上去，直到 html ，那么就是一个绝对路径了。

2.3、布尔值写法：

如果一个人的姓名不是唯一的，身份证号也不是唯一的，但是同时叫张三 并且 身份证号为 123 的人却可以唯一的确定一个人。那么可以这样写：

Xpath = //input[@id='kw1' and @name='wd']

可以 and ， 当然也可以 or :

Xpath = //input[@id='kw1' or @name='wd']

但 or 的实际意义不太。我们一般不需要说，找的人名字或者叫张三，或者身份证号是 123 也可以。

Robot framework 中的写法：

input text	xpath = //*[@id='kw1']	robot framework 学习
input text	xpath = //span[@class='bg s_ipt_w']/input	robot framework 学习
input text	xpath = //input[@id='kw1' and @name='wd']	robot framework 学习

5.2.4 css 定位

CSS(Cascading Style Sheets)是一种语言，它被用来描述 HTML 和 XML 文档的表现。CSS 使用选择器来为页面元素绑定属性。这些选择器可以被 selenium 用作另外的定位策略。

CSS 可以比较灵活选择控件的任意属性，一般情况下定位速度要比 XPath 快，但对于初学者来说比较难以学习使用，下面我们就详细的介绍 CSS 的语法与使用。

CSS 选择器的常见语法：

选择器	例子	描述
.class	.intro	class 选择器，选择 class="intro"的所有元素
#id	#firstname	id 选择器，选择所有 id="firstname"所有元素
*	*	选择所有元素
element	p	元素所有<p>元素
element > element	div > input	选择父元素为 元素的所有 <input> 元素

选择器	例子	描述
.class	.intro	class 选择器，选择 class="intro"的所有元素
#id	#firstname	id 选择器，选择所有 id="firstname"所有元素
*	*	选择所有元素
element	p	元素所有<p>元素
element > element	div > input	选择父元素为 元素的所有 <input> 元素
element + element	div + input	选择紧接在 元素之后的所有 <p> 元素。
[attribute=value]	[target=_blank]	选择 target="_blank" 的所有元素。

通过 class 属性定位：

css=.s_ipt

css=.bg s_btn

csscss_selector()方法用于 CSS 语言定位元素，点号（.）表示通过 class 属性来定位元素。

通过 id 属性定位：

css=#kw

css=#su

井号（#）表示通过 id 属性来定位元素。

通过标签名定位：

css=input

在 CSS 语言中用标签名定位元素不需要任何符号标识，直接使用标签名即可，但我们前面已经了解到标签名重复的概率非常大，所以通过这种方式很难唯一的标识一个元素。

通过父子关系定位：

css=span>input

上面的写法表示有父亲元素，它的标签名叫 span，查找它的所有标签名叫 input 的子元素。

通过属性定位：

css=input[autocomplete='off']

css=input[maxlength='100']

css=input[type='submit']

在 CSS 当中也可以使用元素的任意属性，只要这些属性可以唯一的标识这个元素。

组合定位：

我们当然可以把上面的定位策略组合起来使用，这样就大大加强了元素的唯一性。

css=span.bg s_ipt_wr>input.s_ipt

css=span.bg s_btn_wr>input#su

有一个父元素，它的标签名叫 span，它有一个 class 属性值叫 bg s_ipt_wr，它有一个子元素，标签名叫 input，并且这个子元素的 class 属性值叫 s_ipt。好吧！我们要找的就是具有这么多特征的一个子元素。

Robot framework 中的写法：

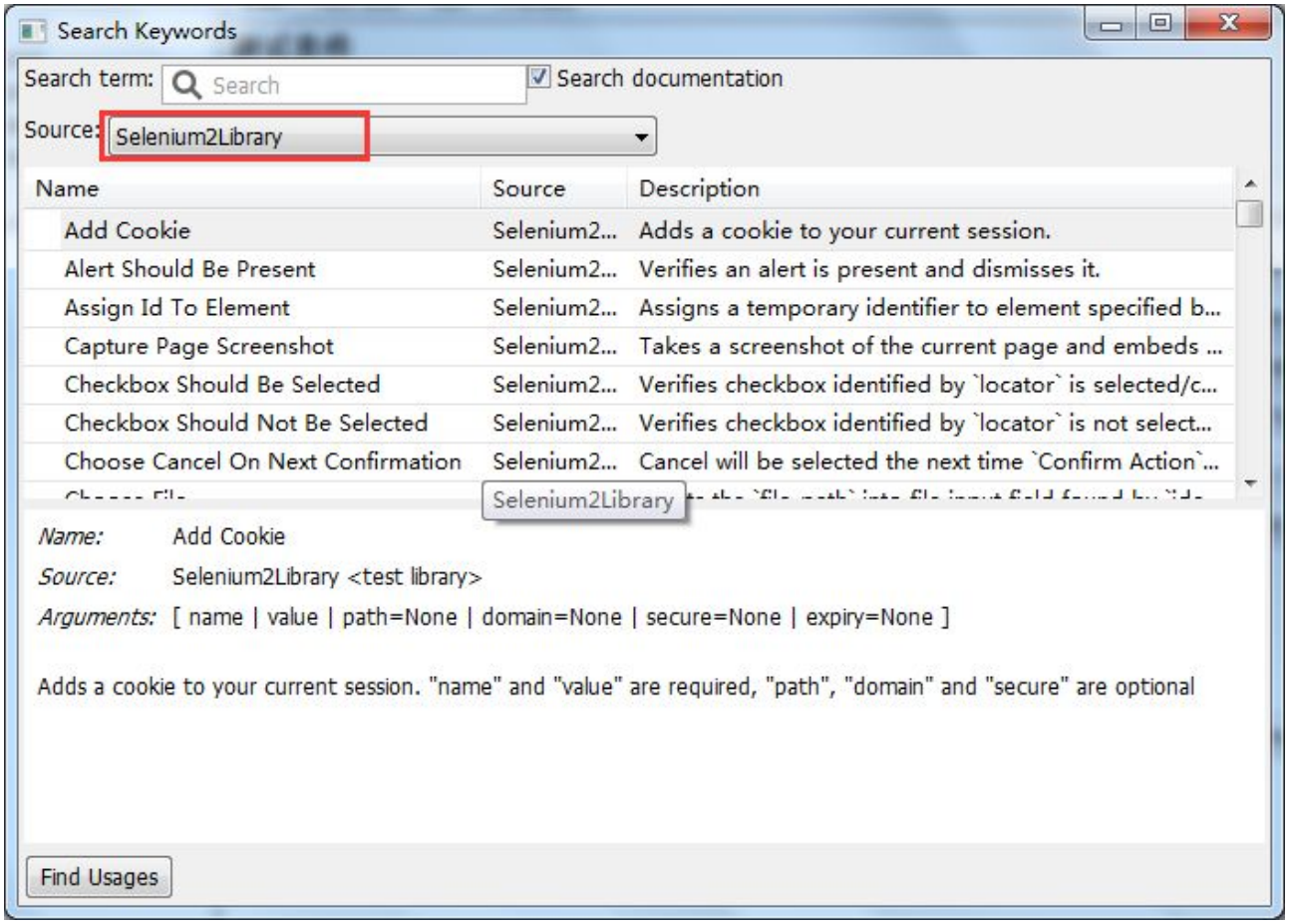
input text	css=#kw	robot framework 学习
input text	css=.s_ipt	robot framework 学习
input text	css=[name=wd]	robot framework 学习

5.3 Selenium2Library 关键字

关于 Selenium2Library 的关键字，我们可以参考：

http://rtomac.github.io/robotframework-selenium2library/doc/Selenium2Library.html#Unselect%20Checkbox

或者通过 F5查找 Selenium2Library 关键字库：



下面就来介绍 Selenium2Library 库中常用的关键字。

5.3.1 浏览器驱动

通过不同的浏览器执行脚本

Open Browser	Htpp://www.xxx.com	chrome

浏览器对应的关键字：

firefox	FireFox
ff	
internetexplorer	Internet Explorer
ie	
googlechrome	Google Chrome
gc	
chrome	
opera	Opera
phantomjs	PhantomJS
htmlunit	HTMLUnit
htmlunitwithjs	HTMLUnit with Javascipt support
android	Android
iphone	Iphone
safari	Safari

open browser 同样也可以打开本地 html 页面，如：

Open Browser	file:///E:/OTCNEWSETT/OTC_qs/js.html	googlechrome

备注：

要想通过不同的浏览打开 URL 地址，一定要安装浏览器相对应的驱动。

chrome 的驱动为：chromedriver.exe 。

IE 的驱动为：IEDriverServer.exe

浏览器默认为空时启动 FireFox。

5.3.2 关闭浏览器

Close Browsers		

Close All Browser		

close browser 关闭当前的浏览器。close all browser 关键所有打开的浏览器和缓存重置。

5.3.3 浏览器最大化

Maximize Browser Window		

Maximize Browser Window 关键字使当前打开的浏览器全屏。

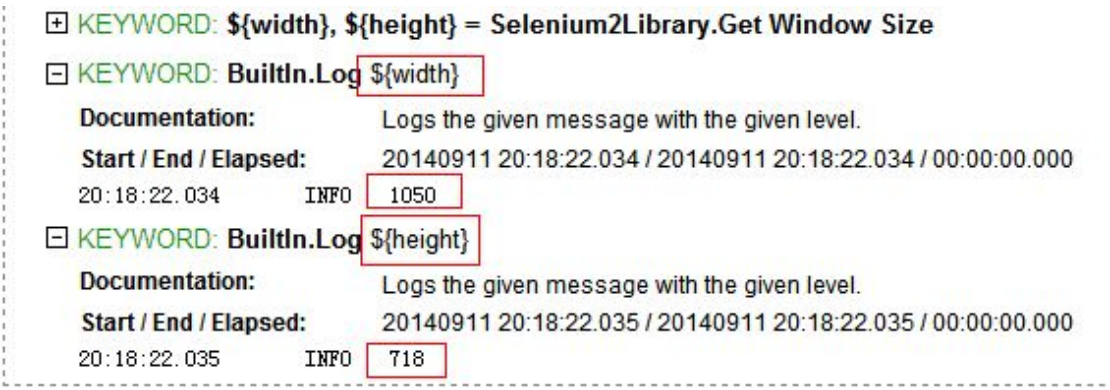
5.3.4 设置浏览器窗口宽、高

Get Window Size	800	600

get windows size 关键字用于打设置打开浏览器的宽度和高度。以像素为单位，第一个参数 800 表示宽度，第二个参数 600 表示高度。

${width}	${height}	get window size
log	${width}	
log	${height}	

get windows size 关键字，用于获取当前浏览器的宽度和高度。获得浏览浏览器窗口宽、高，将显示在 log.html 的日志中。



5.3.5 文本输入

Input Text	xpath=//* [@]	输入信息

input text 关键字用于向文本框内输入内容。

xpath=//* [@] ：表示元素定位，定位文本输入框。

5.3.6 点击元素

Click Element	xpath=//* [@]	

Click Element 关键字用于点击页面上的元素，单击任何可以点击按钮、文字/图片连接、复选框、单选框、甚至是下拉框等。

xpath=//* [@] ：表示元素定位，定位点击的元素。

5.3.7 点击按钮

Click Button	Xpath=//* [@]	

Click Element 关键字用于点击页面上的按钮。

Xpath=//* [@] ：表示元素定位，定位点击的按钮。

5.3.8 等待元素出现

Wait Until Page Contains Element	Xpath=//* [@]	42	error

Wait Until Page Contains Element 关键字用于等待页面上的元素显示出来。

Xpath=//* [@] ：表示元素定位，这里定位出现的元素

42 ： 表示最长等待时间。

Error ： 表示错误提示，自定义错误提示，如：“元素不能正常显示”

5.3.9 获取 title

Get Title		

get title 关键字用于获得当前浏览器窗口的 title 信息。

这里只获取 title 是没有意义的，我们通常会将获取的 title 传递给一个变量，然后与预期结果进行比较。从而判断当前脚本执行成功。

5.3.10 获取 text

Get Text	xpath=//* [@]	

get text 关键字用于获取元素的文本信息。

xpath=//* [@] ： 定位文本信息的元素。

5.3.11 获取元素属性值

Get Element Attribute	id=kw@name	

id=kw@name：id=kw 表示定位的元素。@nam 获取这个元素的 name 属性值。

5.3.12 cookei 处理

get cookies		
get cookie value	Key_name	
add cookie	Key_name	Value_name
delete cookie	Key_name	
delete all cookies		

get cookies 获得当前浏览器的所有 cookie 。

get cookie value 获得 cookie 值。key_name 表示一对 cookie 中 key 的 name 。

add cookie 添加 cookie。添加一对 cooke （key：value）

delete cookie 删除 cookie。删除 key 为 name 的 cookie 信息。

delete all cookies 删除当前浏览器的所有 cookies。

5.3.13 验证

获得浏览器 title 进行比较。

open browser	http://www.baidu.com	chrome
${title}	Get Title	
should contain	${title}	百度一下，你就知道

Open Browser 通过 chrome 打开百度首页。

Get Title 获得浏览器窗口的 titile ，并赋值给变量${title}

Should Contain 比较${title}是否等于“百度一下，你就知道”。

如果 item1 不包含 item2 一次或多次，那么失败。

获得文本信息进行比较

${text}	Get Text	
should contain	${text}	百度一下，你就知道

5.3.14 表单嵌套

有时候和页面中会出现表单嵌套，这个时候需要进入到表单才能操作相关元素。

Select Frame	Xpath=//* [@]	
Unselect Frame		

Select Frame 进入表单，Xpath=//* [@] 表示定位要进入的表单。

Unselect Frame 退出表单。

5.3.15 下拉框选择

Unselect From List By Value	Xpath=//* [@]	vlaue

Unselect From List By Value 关键字用天选择下拉框。

Xpath=//* [@] 定位下拉框；

Vlaue 选择下拉框里的属性值。

5.3.16 执行 JavaScript

在一些特殊的情况下需要调用 JavaScript 代码。

Execute Javascript	$("#tooltip").fadeOut();	

Execute Javascript 关键字用于使用 JavaScript 代码

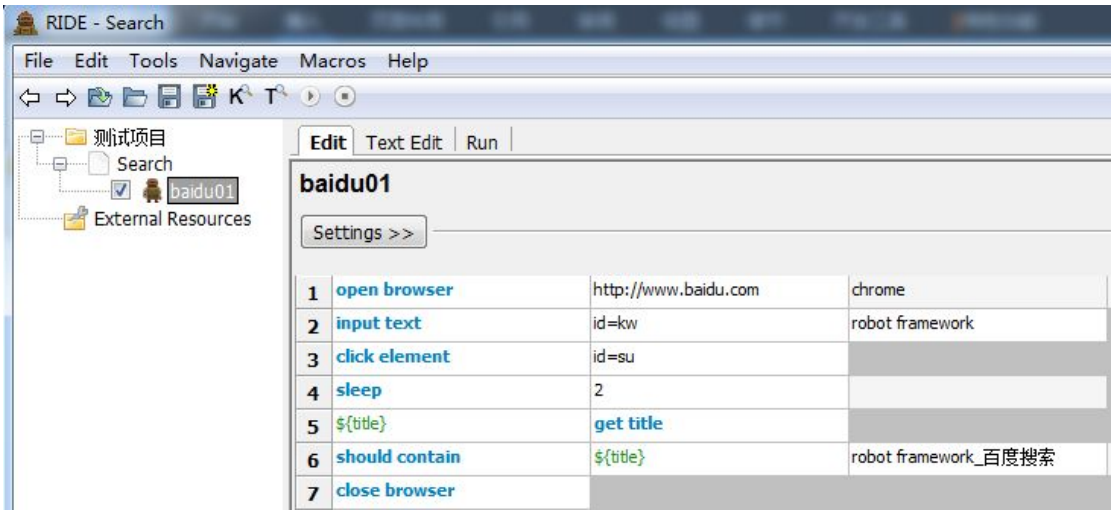
5.4 Robot Framework 分层设计

谈到 Robot Framework 分层的思想，就不得不提“关键字驱动”。

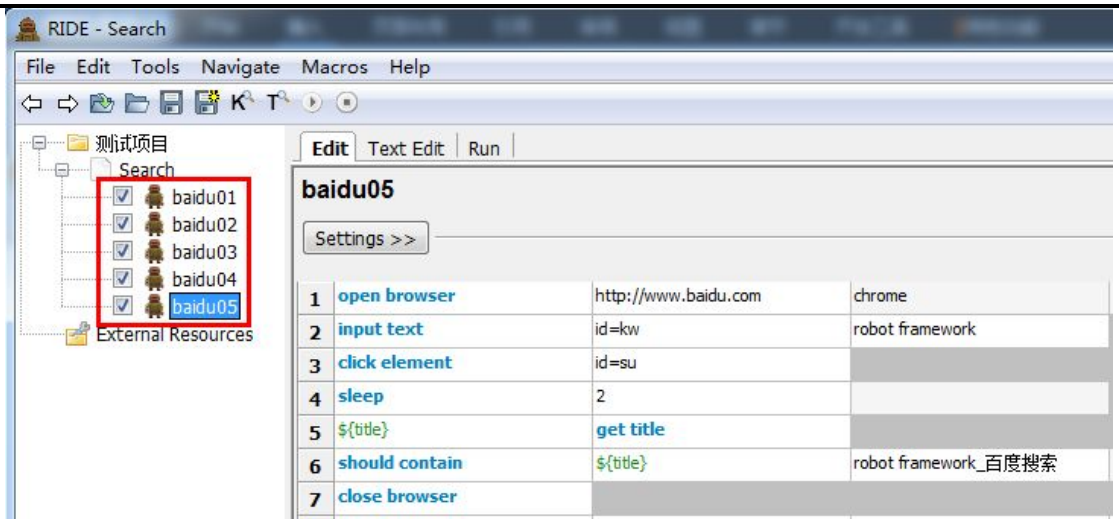
回到分层的思想上，在程序设计的讲究设计模式，设计模式其实就是根据需求使用抽象与封装，其实就是分层思想。把一个实现过程分成不同多层。提高的灵活性，从而达到可扩展性和可维护性。

再回到自动化的话题上，我们可以把操作步骤封装一个一个的方法（关键字），通过调用关键字来实现测试用例。

参考前面创建的一条百度搜索的测试用例。



我现在要写五条百度搜索的用例：



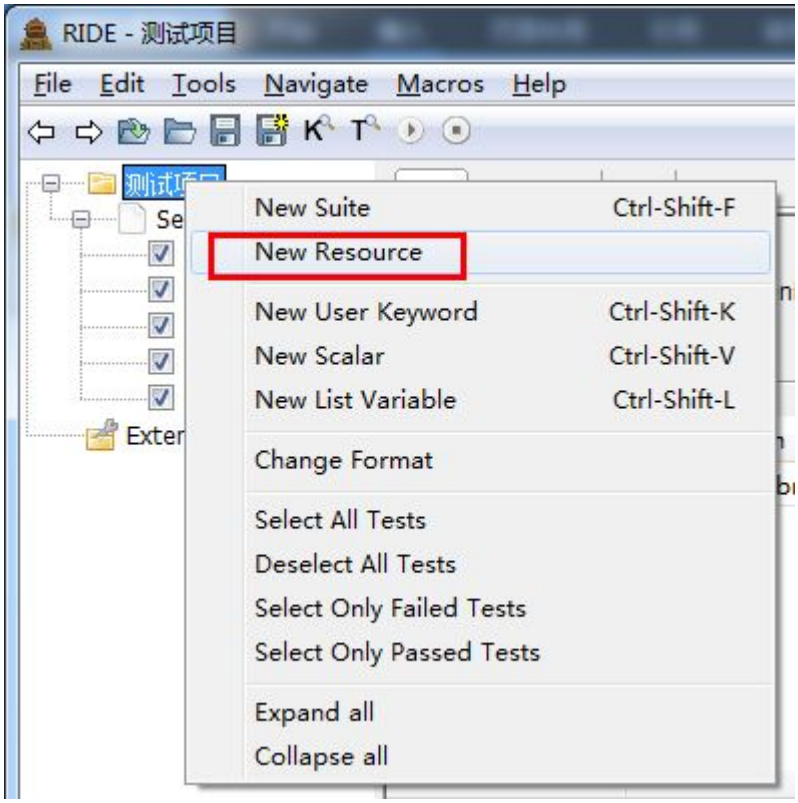
可以在 Search 测试套件下创建 5 条测试用例。其实对于每一条测试用例来说，只是搜索的内容不同，脚本步骤是完全一样的。这样做无疑增加的脚本的冗余，而且不便于维护。假如，百度输入框的定位方式变了，我不得不打开每一条用例进行修改。

我们可以过创建关键字的方式，从而实现分层的思想来解决这个问题。

Robot Framework 创建关键字步骤：

1、创建资源

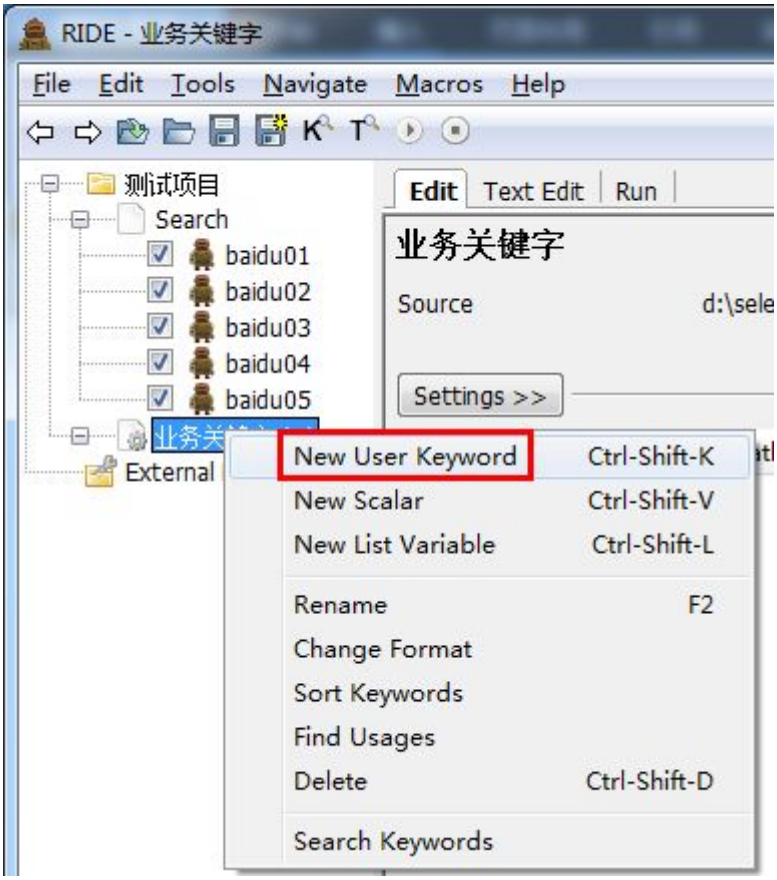
右键“测试项目”选择“new resource”创建资源。



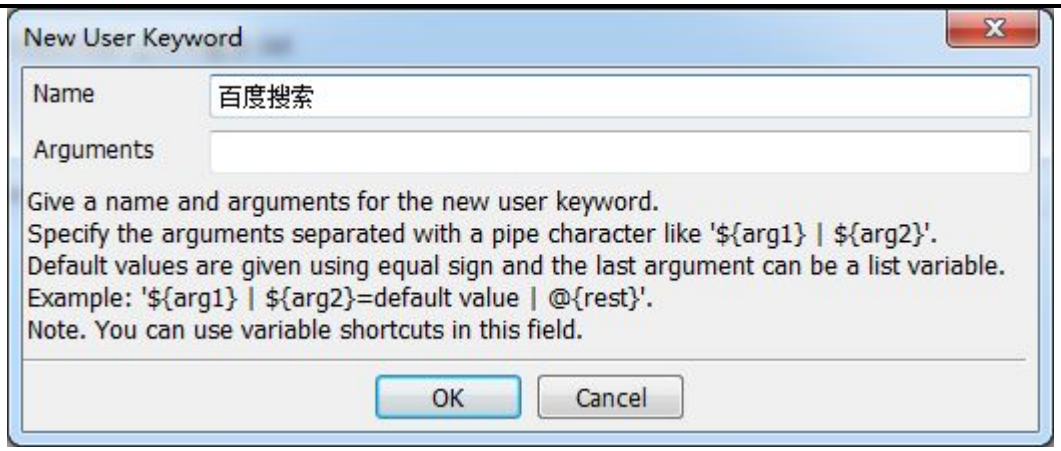
输入资源名称：

2、创建关键字

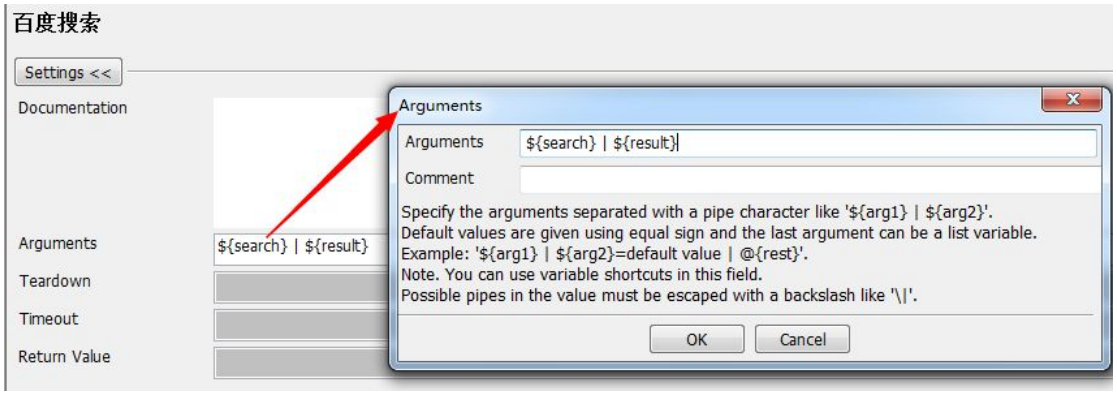
右键“业务关键字”选择“new User Keyword” 来创建用户关键字。



输入关键字的名称：



3、编辑关键字



分析：

对于一个测试用例来说，用户关心的是输入什么内容，得到什么结果。

所以，对于“百度搜索”关键字来说，需要创建两个接口变量${search}和${result} 两个变量，用于接收输入内容和预期结果。

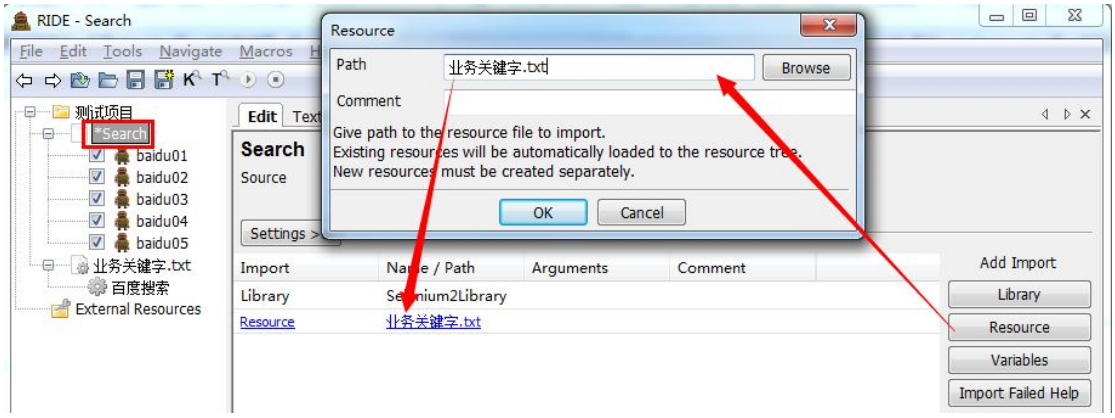
点击 Arguments 输入框，定义变量，多个变量从用“|”隔开。

在百度用户中使用参数化变量。

1	**open browser**	http://www.baidu.com	chrome
2	**input text**	id=kw	${search}
3	**click element**	id=su	
4	**sleep**	2	
5	${title}	**get title**	
6	**should contain**	${title}	${result}
7	**close browser**		
8			

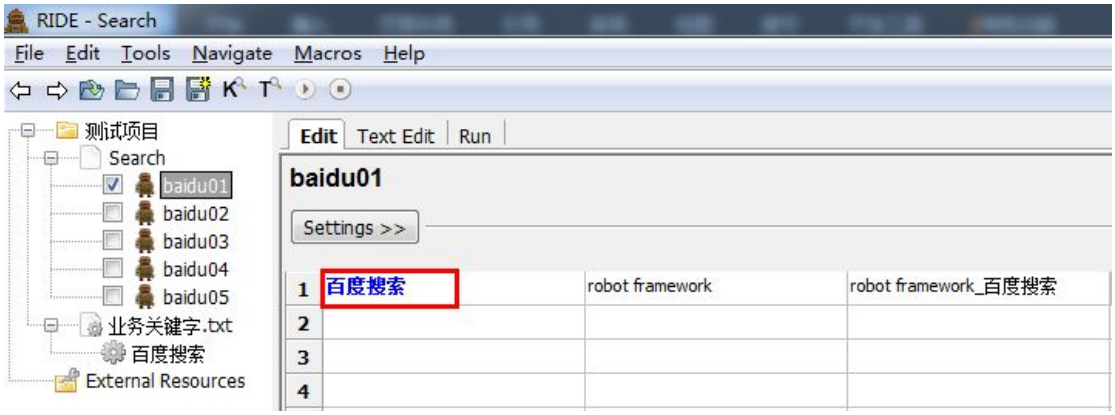
4、添加创建的资源

切换到测试套件（Search）页面，添加资源（业务关键字.txt）



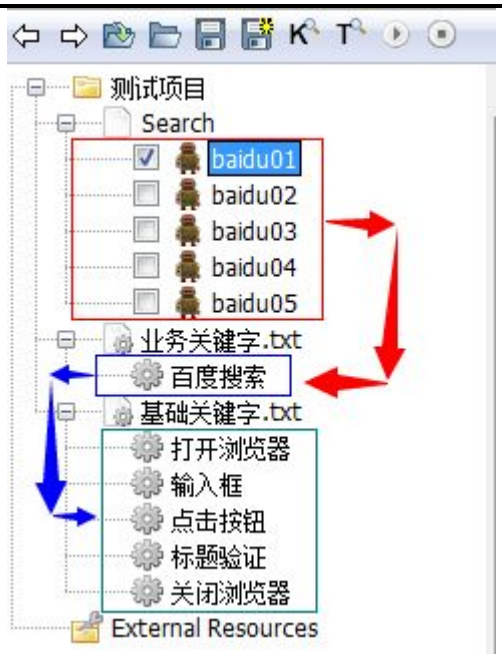
5、调用关键字

现在就可以在测试用例中使用创建的关键字了（百度搜索）。



对于每一条用例来说，调用“百度搜索”关键字，输入搜索内容，输入预期结果即可。不同关心用例是如何执行的。如果百度输入框的定位发生了变化，只用去修改“百度搜索”关键字即可，不用对每一条用例做任何修改。大大提高的用例的维护性和扩展性。

继续分层的设计：



到此，Robot Framework +Selenium 自动化测试粗犷的讲完了。当然还有更多 API 的使用，和细枝末节的设置没有介绍。但我们已经可以拿它来开展自动化工作了。

第 6 章 AutoItLibrary 库

AutoIt 这是一个使用类似 BASIC 脚本语言的免费软件,它设计用于 Windows GUI(图形用户界面)中进行自动化操作。它利用模拟键盘按键，鼠标移动和窗口/控件的组合来实现自动化任务。而这是其它语言不可能做到或无可靠方法实现的(例如 VBScript 和 SendKeys)。

AutoItLibrary 是基于 AutoIt 针对于 Robot Framework 开发的一个库。

6.1 安装 AutoItLibrary 库

1、安装 AutoItLibrary 库

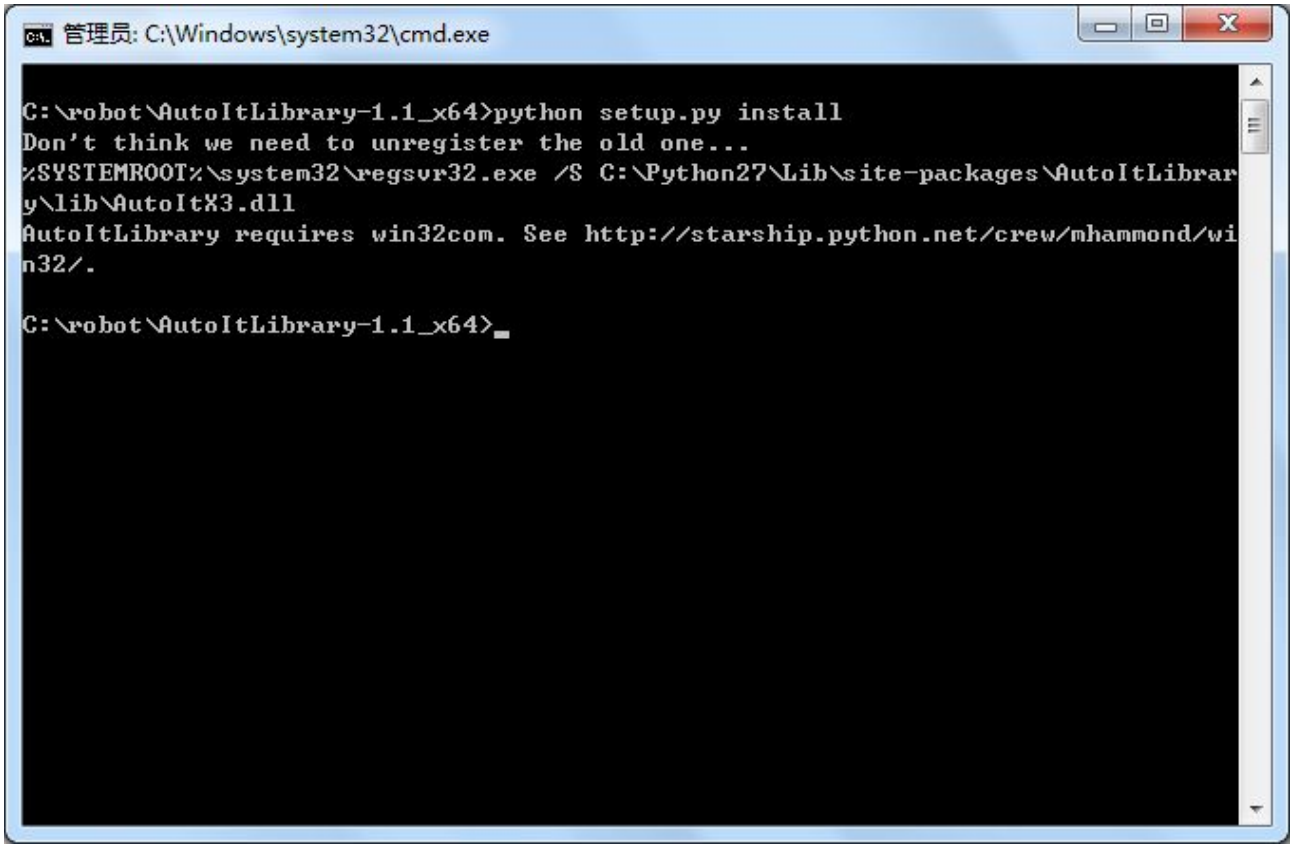
AutoItLibrary 官方地址：

http://code.google.com/p/robotframework-autoitlibrary/

由于 google 网站很难访问，所以我们可以从 CSDN 找到 AutoItLibrary 库的下载。

http://download.csdn.net/detail/liuheng123456/6236097

由于我的环境是 win7 64 位，所以下载：AutoItLibrary-1.1_x64 包，解压执行 setup.py 文件，如下图：



根据上面的提示：AutoItLibrary 库的安装需要 win32com。所以我们需要安装 pywin32。Windows

Pywin32 允许你像 VC 一样的形式来使用 PYTHON 开发 win32 应用

2、安装 pywin32 库

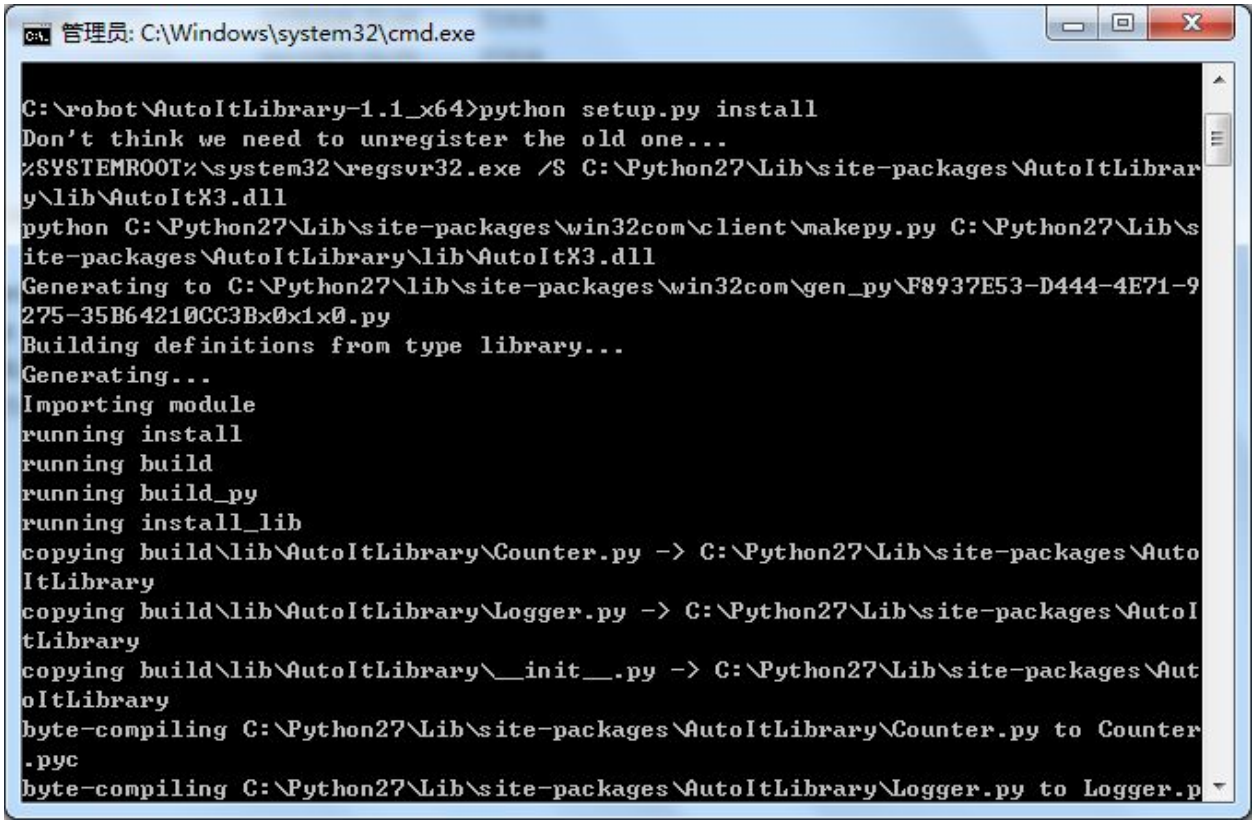
下载 pywin32：

http://sourceforge.net/projects/pywin32/files/

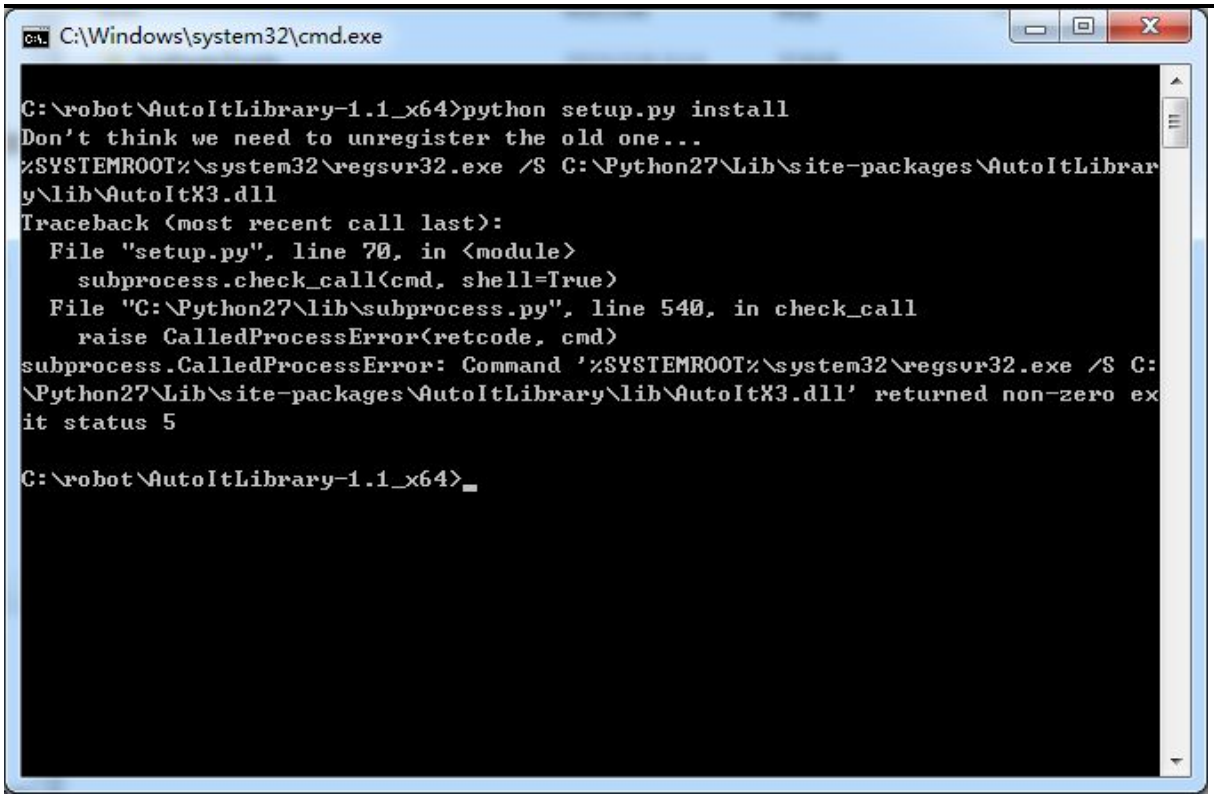
根据自己的系统以 Python 版本下载对应的版本，我的系统为 win7 64、Python2.7，所以下载为：

pywin32-219.win-amd64-py2.7.exe

双击安装，完成之后再来安装 AutoItLbrary 就可以正常安装。



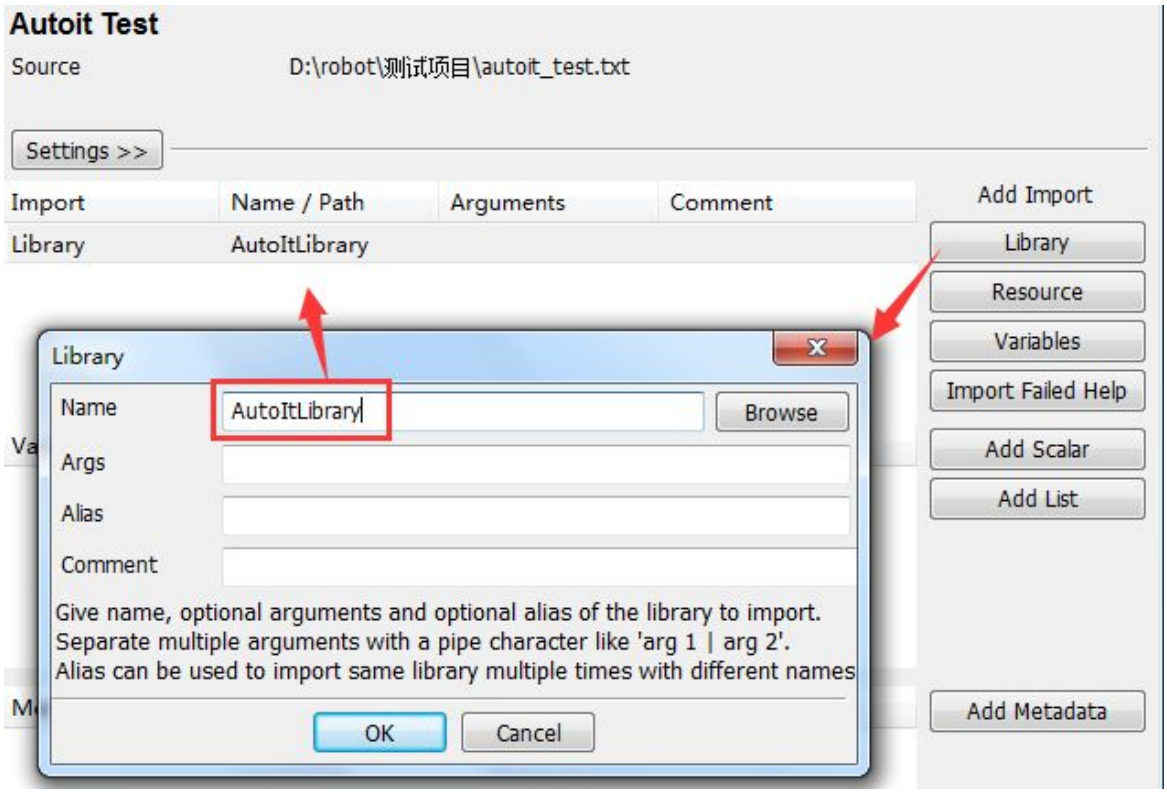
如果在安装的过程中报以下错误：



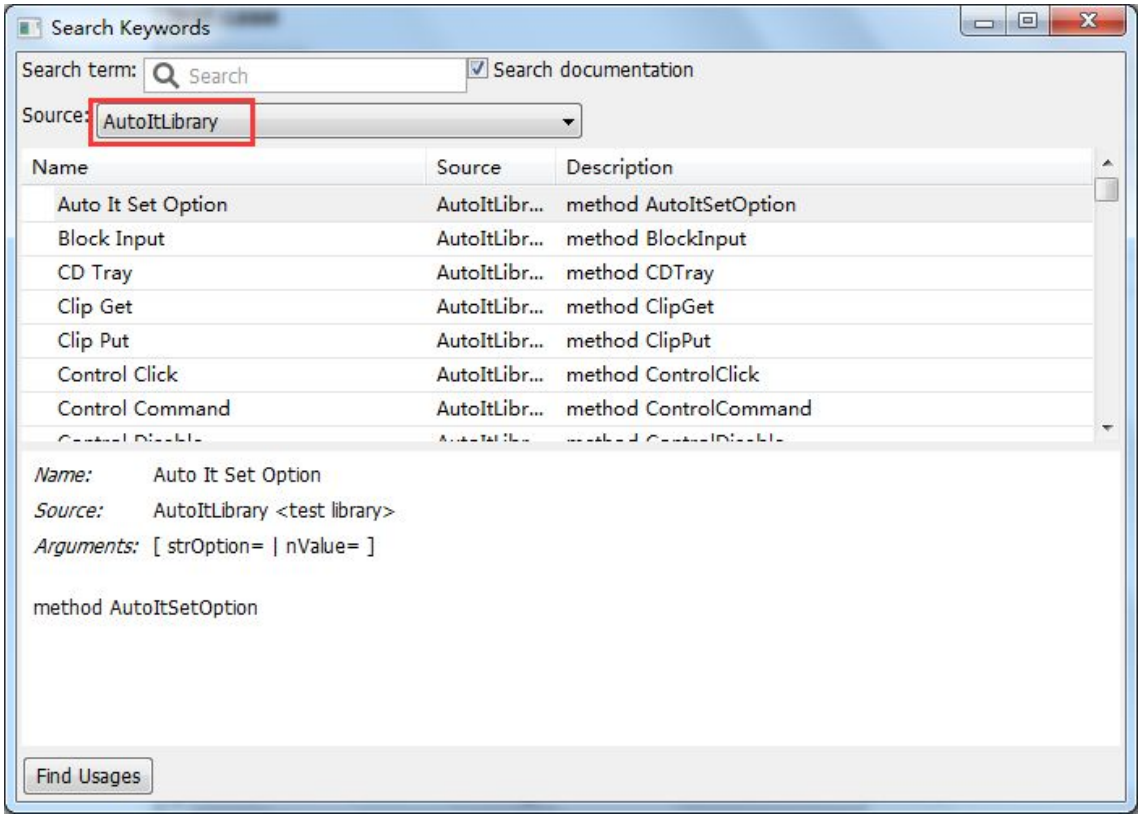
请切换到“Adminstrator”管理员用户再来执行安装。

2、导入 AutoItLibrary 库

打开 RIDE，创建 Autoit_Test 测试套件，导入 AutoItLbrary 库：



通过 F5 查看 AutoItLibrary 库所提供的关键字：



6.2 AutoIt v3 入门

如果你完全不了解 AutoIt 工具的话，那么 AutoItLibrary 库中关键字的使用，相信你很难理解和掌握。所以，我们有必要花些时间来学习 AutoIt。

AutoIt v3 是用以编写并生成具有 BASIC 语言风格的脚本程序的免费软件，它被设计用来在 Windows GUI（用户界面）中进行自动操作。通过它可以组合使用模拟键击、鼠标移动和窗口/控件操作等来实现自动化任务，而这是其它语言所无法做到或尚无可靠方法实现的（比如 VBScript 和 SendKeys）。

AutoIt 最初是为 PC（个人电脑）的“批量处理”而设计，用于对数千台 PC 进行（同样的）配置，不过随着 v3 版本的到来它也很适合用于家庭自动化和编写用以完成重复性任务的脚本。

AutoIt 可以做的事:

- 运行 Windows 及 DOS 下的可执行文件
- 模拟键击动作（支持大多数的键盘布局）
- 模拟鼠标移动和点击动作。
- 对窗口进行移动、调整大小和其它操作。
- 直接与窗口的“控件”交互（设置/获取 文字、移动、关闭，等等）

- 配合剪贴板进行剪切/粘贴文本操作
- 对注册表进行操作

不同于 AutoIt v2，新的 v3 版本含有更多的标准语法——类似于 VBScript 和 BASIC——而且现在支持更复杂的表达式、用户函数、循环以及脚本编写老手们所期待的其它所有内容。

6.2.1 下载与安装

AutoIt v3 下载地址：https://www.autoitscript.com/site/

从网站上下载 AutoIt 并安装，安装完成在菜单中会看到图 4.13 的目录：

图 4.13 AutoIt 菜单

AutoIt Windows Info 用于帮助我们识 Windows 控件信息。

Compile Script to.exe 用于将 AutoIt 生成 exe 执行文件。

Run Script 用于执行 AutoIt 脚本。

SciTE Script Editor 用于编写 AutoIt 脚本。

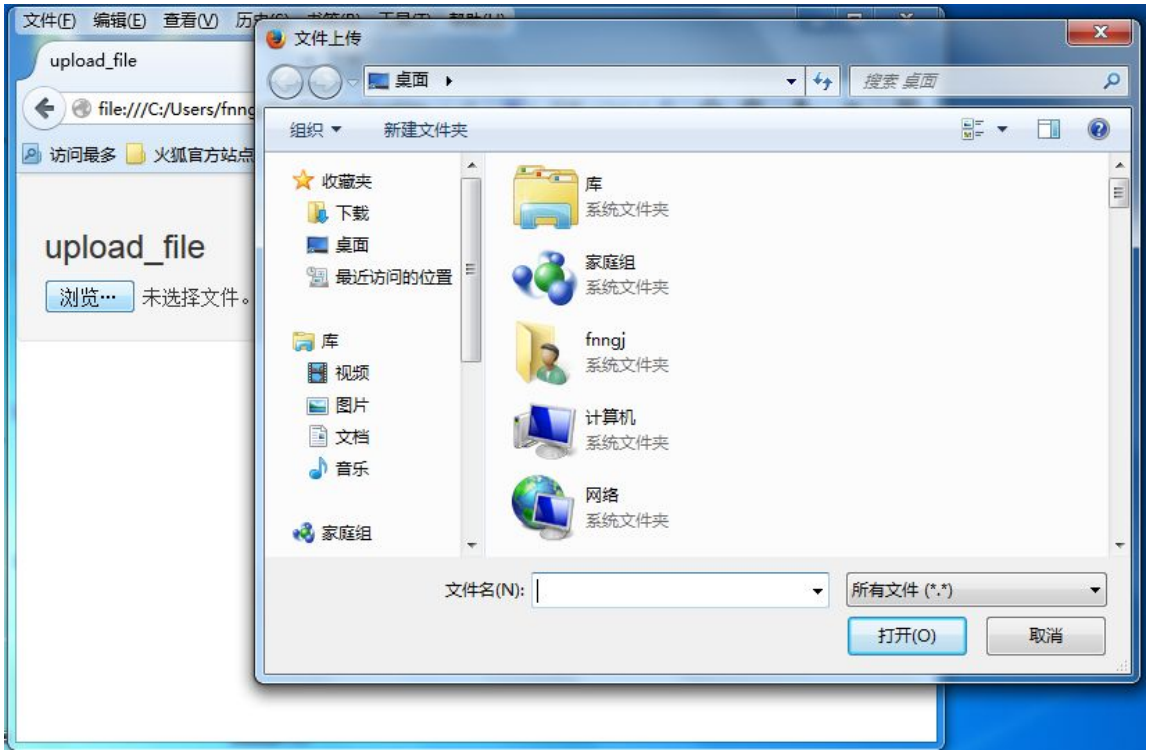
6.2.1 实现 web 上传

这里我们以操作 web 上传窗口为例来介绍 AutoIt v3 的使用。

upfile.html

```
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<title>upload_file</title>
<link href="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.css"
rel="stylesheet" />
</head>
<body>
  <div class="row-fluid">
        <div class="span6 well">
        <h3>upload_file</h3>
          <input type="file" name="file" />
        </div>
  </div>
</body>
<script
src="http://cdn.bootcss.com/bootstrap/3.3.0/css/bootstrap.min.js"></script>
</html>
```

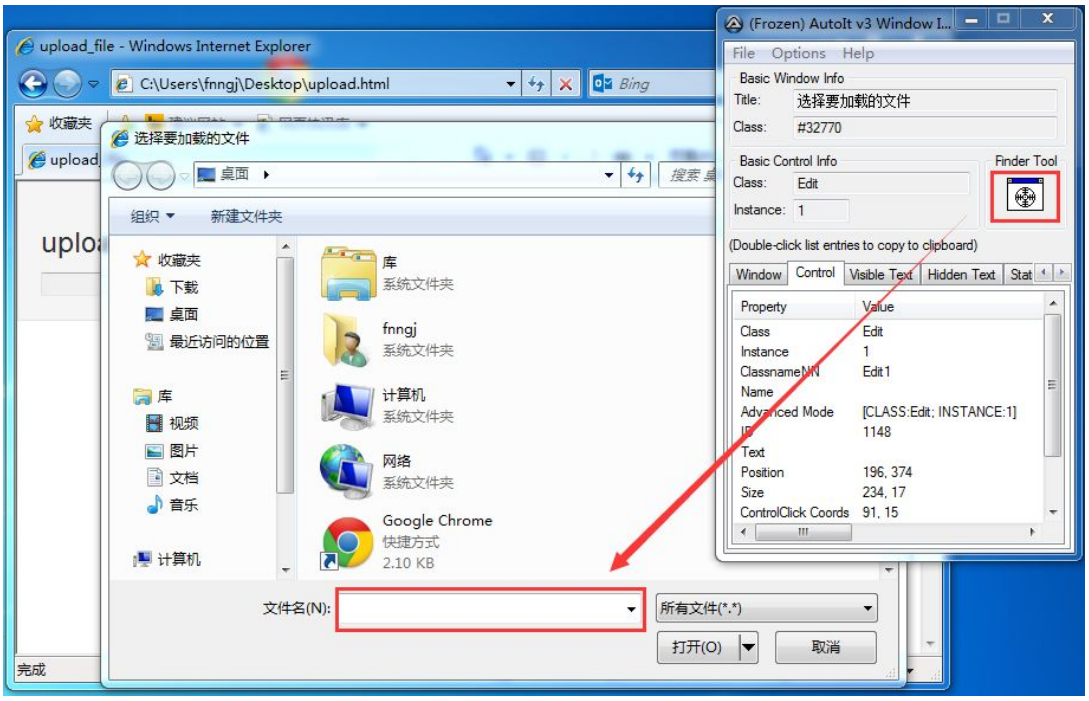
通过浏览器打 upfile.html 文件，效果如下图

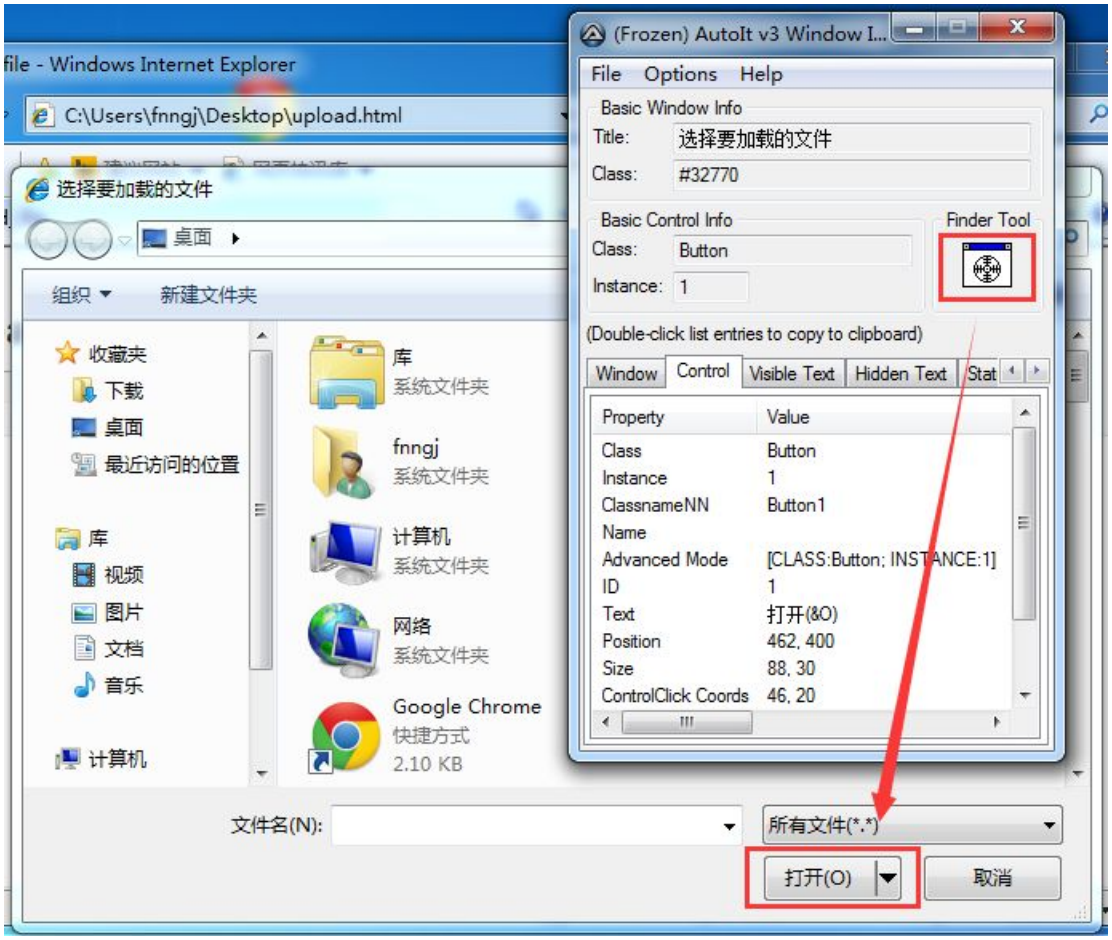


对于 web 页面上打开的本地有 Windows 上传窗口，Selenium 是无法进行识别和定位。

下面以操作 upload.html 上传弹出的窗口为例讲解 AutoIt 实现上传过程。

1、首先打开 AutoIt Windows Info 工具，鼠标点击 Finder Tool，鼠标将变成一个小风扇形状的图标，按住鼠标左键拖动到需要识别的控件上。





如上面的两张图，通过 AutoIt Windows Info 获得以下信息。

窗口的 title 为“选择要加载的文件”，标题的 Class 为“#32770”。

文件名输入框的 class 为“Edit”，Instance 为“1”，所以 ClassnameNN 为“Edit1”。

打开按钮的 class 为“Button”，Instance 为“1”，所以 ClassnameNN 为“Button1”。

2、根据 AutoIt Windows Info 所识别到的控件信息打开 SciTE Script Editor 编辑器，编写脚本。

upfile.au3

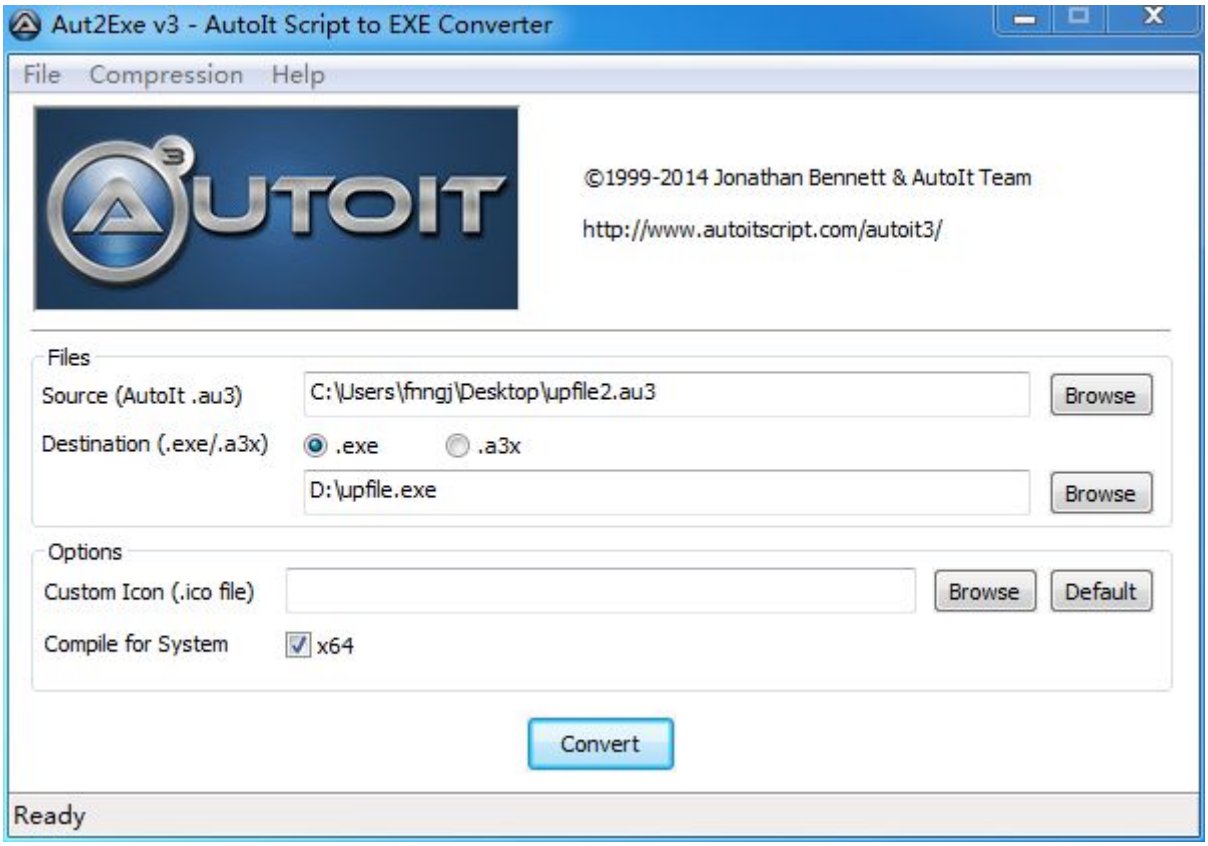
```
;ControlFocus("title","text",controlID) Edit1=Edit instance 1
ControlFocus("选择要加载的文件", "","Edit1")

; Wait 10 seconds for the Upload window to appear
  WinWait("[CLASS:#32770]","",10)

; Set the File name text on the Edit field
  ControlSetText("选择要加载的文件", "", "Edit1", "D:\\upload_file.txt")
  Sleep(2000)

; Click on the Open button
```

```
ControlClick("选择要加载的文件", "","Button1");
```

ControlFocus()方法用于识别 Window 窗口。WinWait()设置 10 秒钟用于等待窗口的显示，其用法与 WebDriver 所提供的 implicitly_wait()类似。ControlSetText()用于向“文件名”输入框内输入本地文件的路径。这里的 Sleep()方法与 Python 中 time 模块提供的 Sleep()方法用法一样，不过它是以毫秒为单位，Sleep(2000)表示固定休眠 2000 毫秒。ControlClick()用于点击上传窗口中的“打开”按钮。

AutoIt 的脚本已经写好了，可以通过菜单栏“Tools”-->“Go” （或按键盘 F5）来运行一个脚本吧！注意在运行时上传窗口当前处于打开状态。

3、脚本运行正常，将其保存为 upfile.au3，这里保存的脚本可以通过 Run Script 工具将其打开运行，但我们的目的是希望这个脚本被 Python 程序调用，那么就需要将其生成 exe 程序。打开 Compile Script to.exe 工具，将其生成为 exe 可执行文件。如图 4.16，



点击“Browse”选择 upfile.au3 文件，点击“Convert”按钮将其生成为 upfile.exe 程序。

4、通过 Robot Framework 加 Selenium2Library 库实现 Web 上传：

open browser	file:///C:/Users/fnngj/Desktop/upload.html	

click element	xpath=/html/body/div/div/input	#点击浏览按钮
sleep	2	Value_name
Evaluate	os.system("d:\\upfile.exe")	#执行 AutoIt 脚本
close browser		

在这个过程中最主要的是学会了如何通过“AutoIt Windows Info”工具来获取控件的属性。

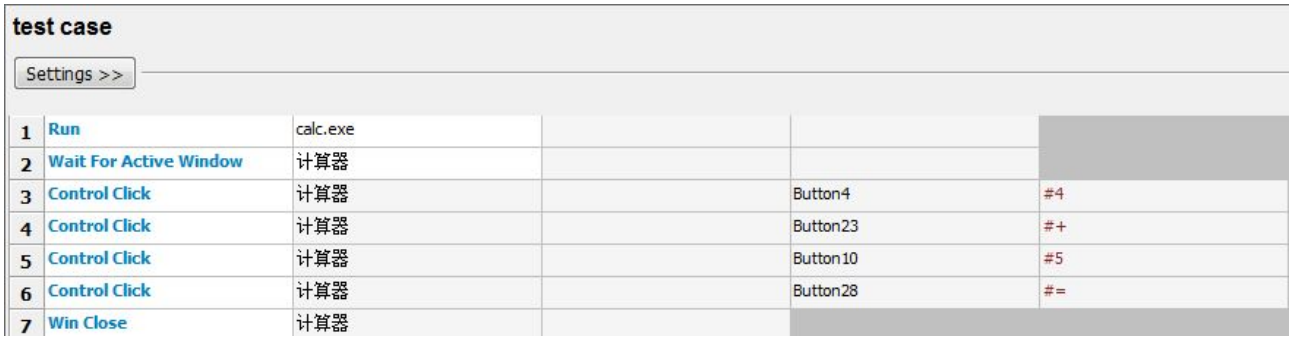
6.3 AutoItLibrary 库

6.3.1 操作计算器的例子

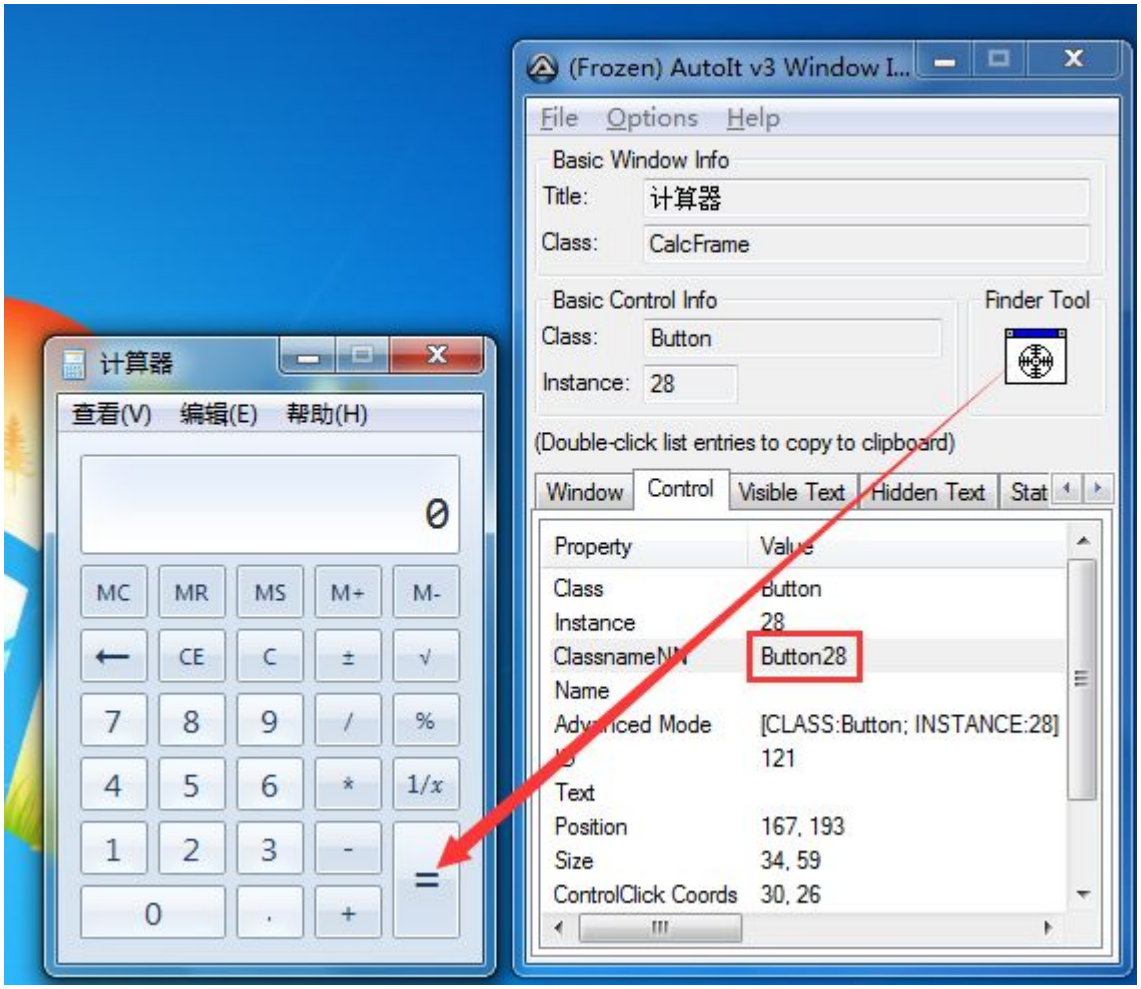
我们以 Windows 自带的计算器的为例，来使用 AutoItLibrary 库。创建 AutoIt 测试用例，编写脚本如下：

Run	calc.exe			
Wait For Active Window	计算器			
Control Click	计算器		Button4	#4
Control Click	计算器		Button23	#+
Control Click	计算器		Button10	#5
Control Click	计算器		Button28	#=
Win Close	计算器			

以上脚本表示，打开计算器，依次点击了计算器上的 “4”、“+”、“5”、“=”等几个按钮。



你一定好奇，为什么“4”的 ClassnameNN 为：Button4，“=”号的的 ClassnameNN 为：Button28。我们通过 AutoIt 自带的 AutoIt Windows Info 工具识别的，如下图：



接下来我们就来学习 AutoItLibrary 库中带用的关键字。

AutoItLibrary 的对象操作大体上有几大主要部分，Window 操作、Control 操作、Mouse 操作、Process 操作、Run 操作、Reg 操作还有一些其他的操作。

6.3.2 运行程序

Run 关键字用来启动程序。

Run	FileName	WorkingDir=	Flag=
Run	calc.exe		
Run	e:/upload.exe		

6.3.3 关闭程序

Win Close 关键字用来关闭程序。

Win Close	strTitle=	strText=
Win Close	计算器	
Win Close	文件上传	

strTitle 是指打开窗口的标题。

6.3.4 控制点击

Control Click 关键字发送控制命令给鼠标点击。

Control Click	strTitle=	strText=	strControl=	strButton=LEFT	nNumClicks=1	nX=-2147483647	nY=-2147483647
Control Click	计算器		Button4				
Control Click	计算器		Button10				

6.3.5 发送

Send 关键字模拟按钮发送到窗口。

Send	strSendText=	nMode=0
Run	calc.exe	
Send	123456	
Send	{F1}	
Send	4{+}5{=}	

“123456”会被输入到计算器的计算框内。

对于非数字的键盘输入用“{}”花括号括起来。“{F1}”表示键盘 F1；“{+}”表示键盘“+”加号；“{=}”表示键盘“=”等号。

组合键 Alt+2：

Send	{ALTDOWN}	
Send	2	
Send	{ALTUP}	

执行{ALTDOWN}表示按下键盘 Alt 键，{ALTUP}表示松开 Alt 键。

组合键 Ctrl+a、Ctrl+x、Ctrl+v：

Send	{CTRLDOWN}	
Send	a	
Send	x	
Send	v	
Send	{CTRLUP}	

执行{CTRLDOWN}表示按下键盘 Ctrl 键，然按 a 键“全选”，按 x 键“剪切”，按 v 键“粘贴”。

6.3.6 等待活动窗口

Wait For Active Window 关键字会等待窗口显示出来。

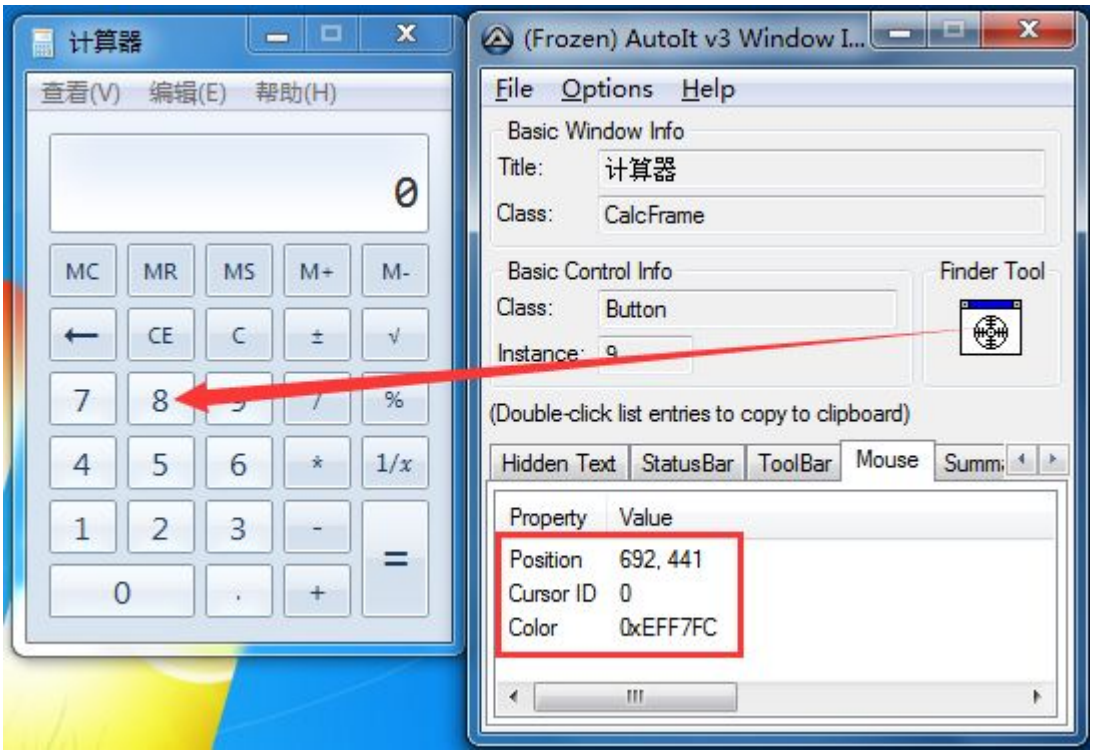
Wait For Active Window	WindowTitle=	WindowText=	TimeOut=-1
Run	calc.exe		
Wait For Active Window	计算器		

6.3.7 鼠标点击

Mouse Click 执行鼠标点击操作。

Mouse Click	strButton=LEFT	nX=-2147483647	nY=-2147483647	nClicks=1	nSpeed=-1
Run	calc.exe				
Mouse Click		692	441		

个人不建议使用这种定位，如果移动计算器窗口，那么上面按钮的坐标也会发生变化。导致定位不准确。获取定位坐标如下。



6.3.8 关闭进程

Process Close 关键字用于关闭进程。

Process Close	strProcess=	nMode=0
Process Close	calc.exe	

6.3.9 获得窗口的宽高

Win Get Client Size Height 关键字用于获取程序窗口的高度。

Win Get Client Size Width 关键字用于获取程序窗口的高度。

Win Get Client Size Height	strTitle=	strText=
Win Get Client Size Width	strTitle=	strText=
run	calc.exe	
Wait For Active Window	计算器	
${heigh}	Win Get Client Size Height	计算器
${width}	Win Get Client Size Width	计算器
log	${heigh}	
log	${width}	

执行结果：

```
Starting test: 测试项目.测试套件.test case
20150311 09:26:02.668 :  INFO : AutoItLibrary.Run(FileName='calc.exe',
WorkingDir='', Flag='')
20150311 09:26:03.231 :  INFO :
AutoItLibrary.WaitForActiveWindow(WindowTitle='\u8ba1\u7b97\u5668',
WindowText='', TimeOut=-1)
20150311 09:26:03.231 :  INFO :
AutoItLibrary.WinWait(WindowTitle='\u8ba1\u7b97\u5668', WindowText='',
TimeOut=60)
20150311 09:26:03.231 :  INFO :
AutoItLibrary.WinWaitActive(WindowTitle='\u8ba1\u7b97\u5668', WindowText='',
TimeOut=60)
20150311 09:26:03.237 :  INFO : ${heigh} = 264
20150311 09:26:03.239 :  INFO : ${width} = 212
20150311 09:26:03.241 :  INFO : 264
20150311 09:26:03.244 :  INFO : 212
Ending test:   测试项目.测试套件.test case
```

6.3.10 窗口标题

Win Get Title　用于获取窗口标题。

Win Set Title　用于设置窗口标题。

Win Get Title	strTitle=	strText=	
Win Set Title	strTitle=	strText=	strNewTitle=
Run	calc.exe		
Win Set Title	计算器		计算机
${tile}	Win Get Title	计算机	
log	${tile}		

Win Set Title 将“计算器”改为“计算机”；Win Get Title 获取当前“计算机”窗口的 title，将通过 log 打印出来。

6.4 帮助

在 AutoItLibrary-1.1_x64 的安装包里的 doc 目录下有一个 AutoItLibrary.html 文件，它包含了 AutoItLibrary 提供的所有关键字。

Shortcuts

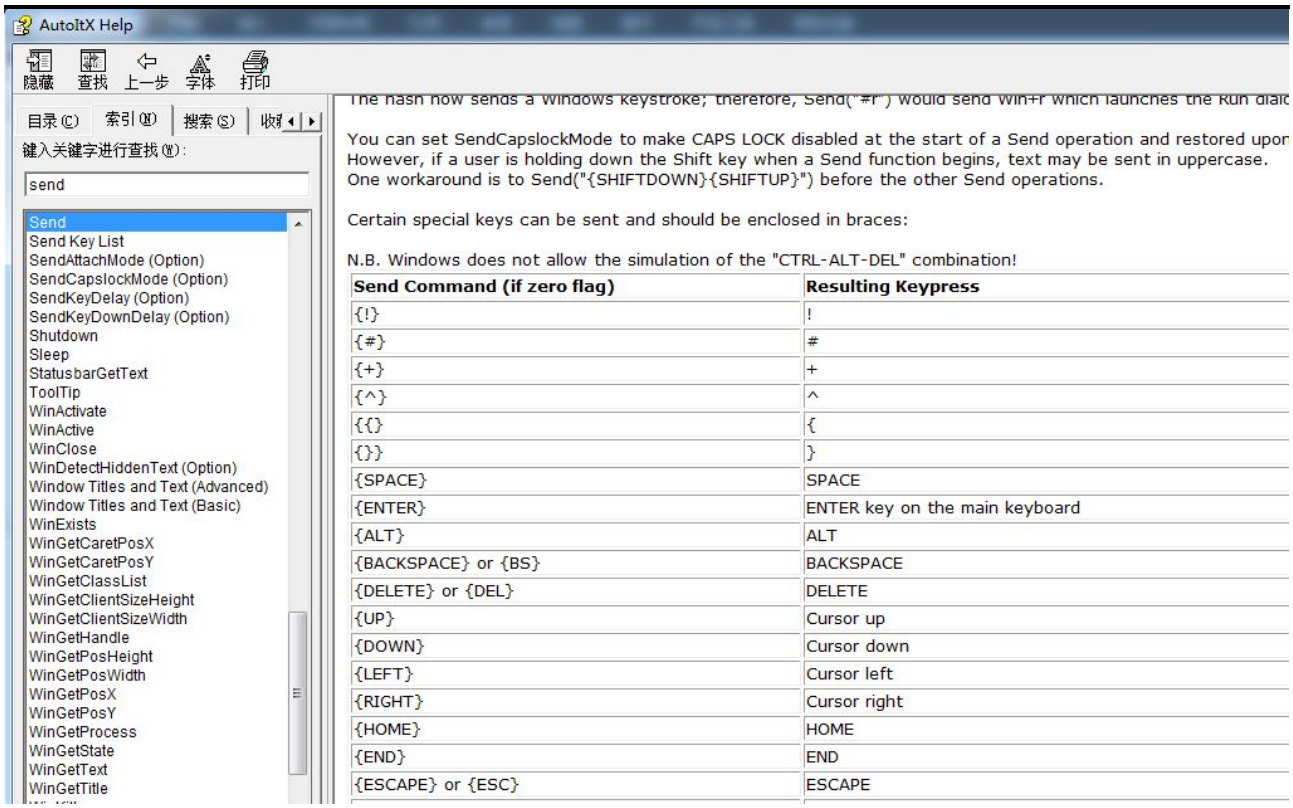
Auto It Set Option Block Input CD Tray Clip Get Clip Put Control Click Control Command
Control Disable Control Enable Control Focus Control Get Focus Control Get Handle
Control Get Pos Height Control Get Pos Width Control Get Pos X Control Get Pos Y
Control Get Text Control Hide Control List View Control Move Control Send Control Set Text
Control Show Control Tree View Drive Map Add Drive Map Del Drive Map Get
Get Active Window Image Get Auto It Version Get Screen Image Get Version Ini Delete Ini Read
Ini Write Init Is Admin Mouse Click Mouse Click Drag Mouse Down Mouse Get Cursor
Mouse Get Pos X Mouse Get Pos Y Mouse Move Mouse Up Mouse Wheel Opt Pixel Checksum
Pixel Get Color Pixel Search Process Close Process Exists Process Set Priority Process Wait
Process Wait Close Reg Delete Key Reg Delete Val Reg Enum Key Reg Enum Val Reg Read Reg Write
Run Run As Set Run Wait Send Shutdown Statusbar Get Text Tool Tip Wait For Active Window
Win Activate Win Active Win Close Win Exists Win Get Caret Pos X Win Get Caret Pos Y
Win Get Class List Win Get Client Size Height Win Get Client Size Width Win Get Handle
Win Get Pos Height Win Get Pos Width Win Get Pos X Win Get Pos Y Win Get Process Win Get State
Win Get Text Win Get Title Win Kill Win List Win Menu Select Item Win Minimize All
Win Minimize All Undo Win Move Win Set On Top Win Set State Win Set Title Win Set Trans
Win Wait Win Wait Active Win Wait Close Win Wait Not Active

Keywords

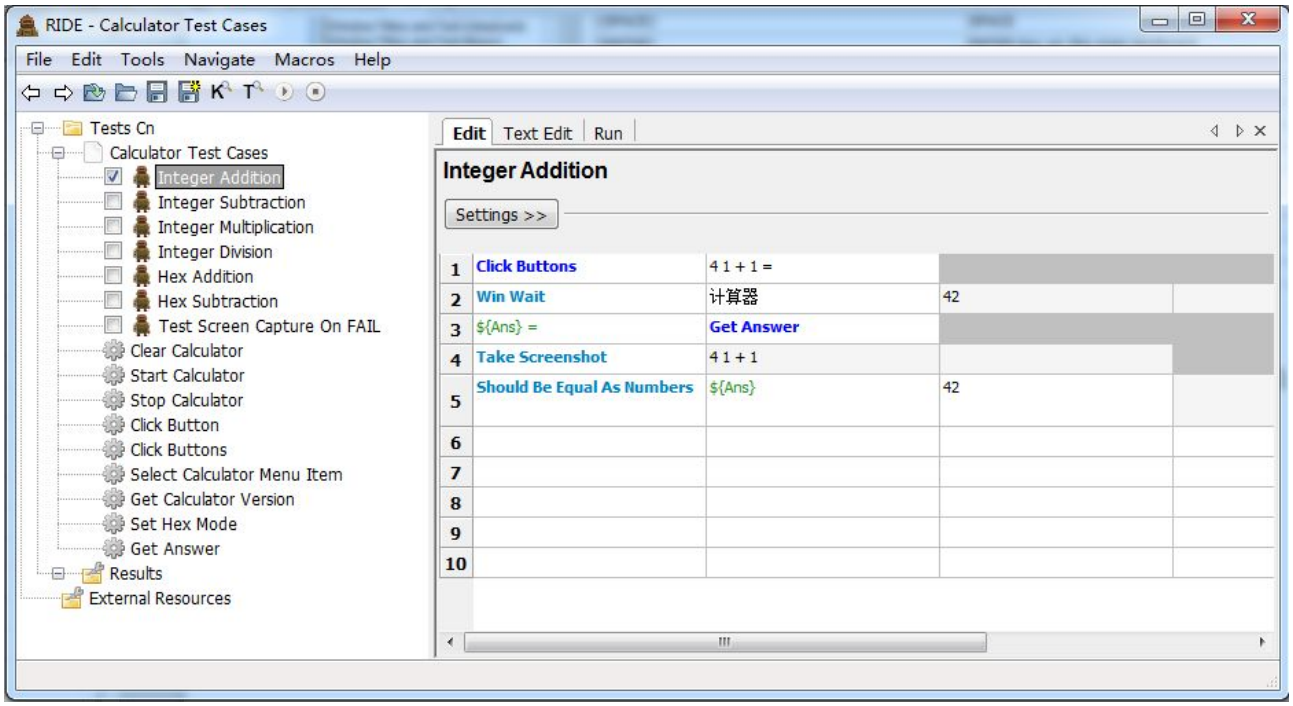
Keyword	Arguments	Documentation
Auto It Set Option	*strOption=, nValue=*	method AutoItSetOption
Block Input	*nFlag=*	method BlockInput
CD Tray	*strDrive=, strAction=*	method CDTray
Clip Get		method ClipGet
Clip Put	*strClip=*	method ClipPut
Control Click	*strTitle=, strText=, strControl=,*	method ControlClick

这份文档只罗列了 AutoItLibrary 库所关键字，但对于关键字的使用并没说明和例子。对此我们可以参考 AutoIt 帮助文档。找到安装包...\AutoItLibrary-1.1_x64\3rdPartyTools\AutoIt\ 目录下的 AutoItX.chm 文件。

例如，AutoItLibrary 库中提供了“Send”关键字，在 AutoItX.chm 中就能打开 send 方法的具体解释和例子。



除此之外，在安装包的...\AutoItLibrary-1.1_x64\tests_cn\还提供了关于计算器的项目。双击“RobotIDE.bat” 文件启动 Robot Framework RIDE 打开项目。



第 7 章 DatabaseLibrary 库

DatabaseLibrary 库用于数据库的操作；这也是常用到的一个测试库。

DatabaseLibrary 用于 Robot Framework 的使用。这可以让你查询你的数据库的行动已取得验证后的结果。它兼容任何数据库 API 规范 2.0 模块。

7.1 安装 DatabaseLibrary 库

DatabaseLibrary 下载地址：

https://pypi.python.org/pypi/robotframework-databaselibrary/0.6

在线文档：

http://franz-see.github.io/Robotframework-Database-Library/

如果像安装普通的 Python 程序，可以下载 tar.gz 文件，解压并运行 setup.py 文件进行安装。

cmd.exe

```
C:\robot\robotframework-databaselibrary-0.6>python setup.py install
```

因为在上一小节中我们已经安装了 pip，所以通过 pip 命令安装更为方便和快捷：

cmd.exe

```
C:\Python27\Lib\site-packages>pip install robotframework-databaselibrary
```

现在只安装 DatabaseLibrary 库，Python 操作不同的数据库，还需要安装相应的数据库驱动。

Oracle 数据库驱动：cx_Oracle

https://pypi.python.org/pypi/cx_Oracle

cx_Oracle 是一个用来连接并操作 Oracle 数据库的 Python 扩展模块， 支持包括 Oracle 9.2 10.2 以及 11.1 等版本。

MySQL 数据库驱动：PyMySQL

https://pypi.python.org/pypi/PyMySQL/

这个包包含一个纯 python MySQL 客户端库。

7.2 操作 Oracle 数据库

这里我们以操作 Oracle 数据库为例，介绍如何操作数据。

7.2.1 连接数据库

连接 Oracle 数据：

Connect To Database Using Custom Params	cx_Oracle	'username','password','192.168.201.138:1521/ORCL'

Connect To Database Using Custom Params ：连接 Oracle 数据库关键字。

cx_Oracle：连接 oracle 驱动。

'username','password','192.168.201.138:1521/ORCL' ：

连接数据库配置信息，用户名，密码，IP 地址，端口号，数据库名。

7.2.2 执行 SQL 语句

执行 sql 语句：

Connect To Database Using Custom Params	cx_Oracle	'username','password','192.168.201.138:1521/ORCL'
Execute Sql String	select * from student	
Disconnect From Database		

Execute Sql String 关键字用于执行 sql 语句。注意 sql 语句结尾不要有分号“；”。

Disconnect From Database 关键字用于断开与数据的连接。

sql 语句结果的输出到测试报告：

Connect To Database Using Custom Params	cx_Oracle	'username','password','192.168.201.138:1521/ORCL'
${result}	Execute Sql String	select * from student
log	${result}	
Disconnect From Database		

7.2.3 执行 SQL 文件

执行 sql 文件：

Connect To Database Using	cx_Oracle	'username','password','192.168.201.138:1

Custom Params		521/ORCL'
Execute Sql Script	${CURDIR}\\test_script\sqlfile.sql	
Disconnect From Database		

Execute Sql Script 关键字用于执行 SQL 文件。

${CURDIR} 表示当前项目路径。

7.2.4 添加系统关键字

在 Database Library 库中所提供的 Execute Sql Script 不支持 sql 脚本文件中包含 begin,end 函数，例如：

```
declare
 custcode NUMBER(19,0) :='102600111';       --客户代码
 username VARCHAR2(32) :='理财_个人1026no111';  -- 用户名
 usertype CHAR(1) :='0';        --0-个人 1-机构
 corporg NUMBER(5) :='999';     --法人机构
 intorg NUMBER(5) :='813';      --内部机构
 idtype CHAR(2) :='10';
 idcode VARCHAR2(48) :='1234567-102612';
 ecifcode VARCHAR2(32) :='102600111';

begin
 delete from USERS where USER_CODE = custcode;     --删除"用户表"中客

 insert into USERS (USER_CODE, USER_ROLES, USER_NAME, USER_TYPE, COR
 values (custcode, '2', username, usertype, corporg, intorg, idtype,

 commit;
 end;
```

通过 Execute Sql Script 关键字执行脚本报错：

```
test_case2                                                                    | FAIL |
DatabaseError: ORA-06550: line 1, column 44:
PLS-00103: Encountered the symbol "end-of-file" when expecting one of the following:

   * & = - + ; < / > at in is mod remainder not rem
   <an exponent (**)> <> or != or ~= >= <= <> and or like like2
   like4 likec between || multiset member submultiset
```

那么就需要我们自己添加关键字了。

找到....\Python27\Lib\site-packages\DatabaseLibrary\ 目录下的 query.py 文件。

创建 execute_sql_funcfile 函数（关键字）：

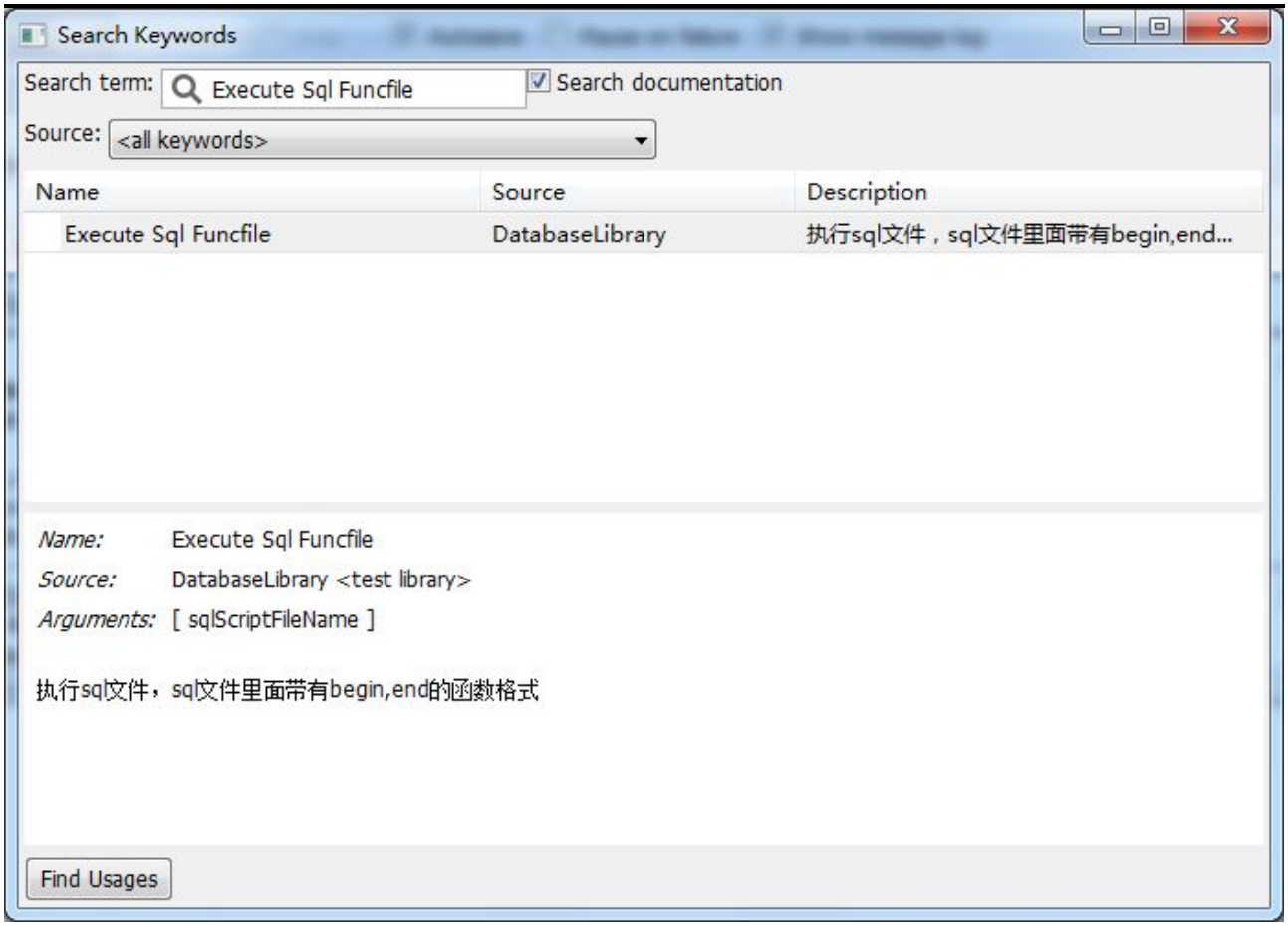
query.py

……

```
    def execute_sql_funcfile(self, sqlScriptFileName):
        """
        执行 sql 文件, sql 文件里面带有 begin,end 的函数格式
        """
        sqlScriptFile = open(sqlScriptFileName)
        cur = None
        try:
            cur = self._dbconnection.cursor()
            sqlStatement = ''
            for line in sqlScriptFile:
                line = line.strip()
                if (line.startswith('/*')==1 and line.endswith('*/')==1):
                    continue
                if line.startswith('#'):
                    continue
                if line.startswith('--'):
                    continue
                if (line.startswith('--')==0 and line.find('--')!=-1):
                    line =line[:line.find('--')]
                if (line == ''):
                    continue
                sqlStatement += line + ' '
            sqlStatement = sqlStatement.replace('\n',' ')
            print  sqlStatement
            if len(sqlStatement) != 0:
                self.__execute_sql(cur, sqlStatement)

            self._dbconnection.commit()
        finally:
            if cur :
                self._dbconnection.rollback()
```

……

然后，在 robot framework 中 F5 搜索 Execute Sql Funcfile 就可以找到我们自定义的关键字了。



执行 SQL 脚本如下：

Connect To Database Using Custom Params	cx_Oracle	otcnewsett','otcnewsett','192.168.201.138:1521/ORCL'
Execute Sql Funcfile	{CURDIR}\\test_script\sqlfile.sql	
Disconnect From Database		

第 8 章 系统关键字开发

其实我的需求也非常简单，接收一个目录路径，自动遍历目录下以及子目录下的所有批处理（.bat）文件并执行。

首先在..\Python27\Lib\site-packages 目录下创建 CustomLibrary 目录，用于放自定义的 library 库。在其下面创建 runbat.py 文件：

runbat.py

```
#-*- coding:utf-8 -*-
'''
    created by bugmaster 2015-01-29
'''
__version__ = '0.1'
from robot.api import loggerimport os
class Runbat(object):

    def run_all_bat(self,path):
        u'''接收一个目录的路径，并执行目录下的所有 bat 文件.例
         | run all bat                | filepath                |
        '''
        for root,dirs,files in os.walk(path):
            for f in files:
                if os.path.splitext(f)[1] == '.bat':
                    os.chdir(root)
                    #print root,f
                    os.system(f)

    def __execute_sql(self, path):
        logger.debug("Executing : %s" % path)
        print path

    def decode(self,customerstr):
        return customerstr.decode('utf-8')
```

```
if __name__ == "__main__":
    path = u'D:\\test_boject'
    run = Runbat()
    run.run_all_bat(path)
```

注意在 run_all_bat()方法下面加上清晰的注释，最好给个实例。这样在 robot framework 的帮助中能看到这些信息，便于使用者理解这个关键字的使用。

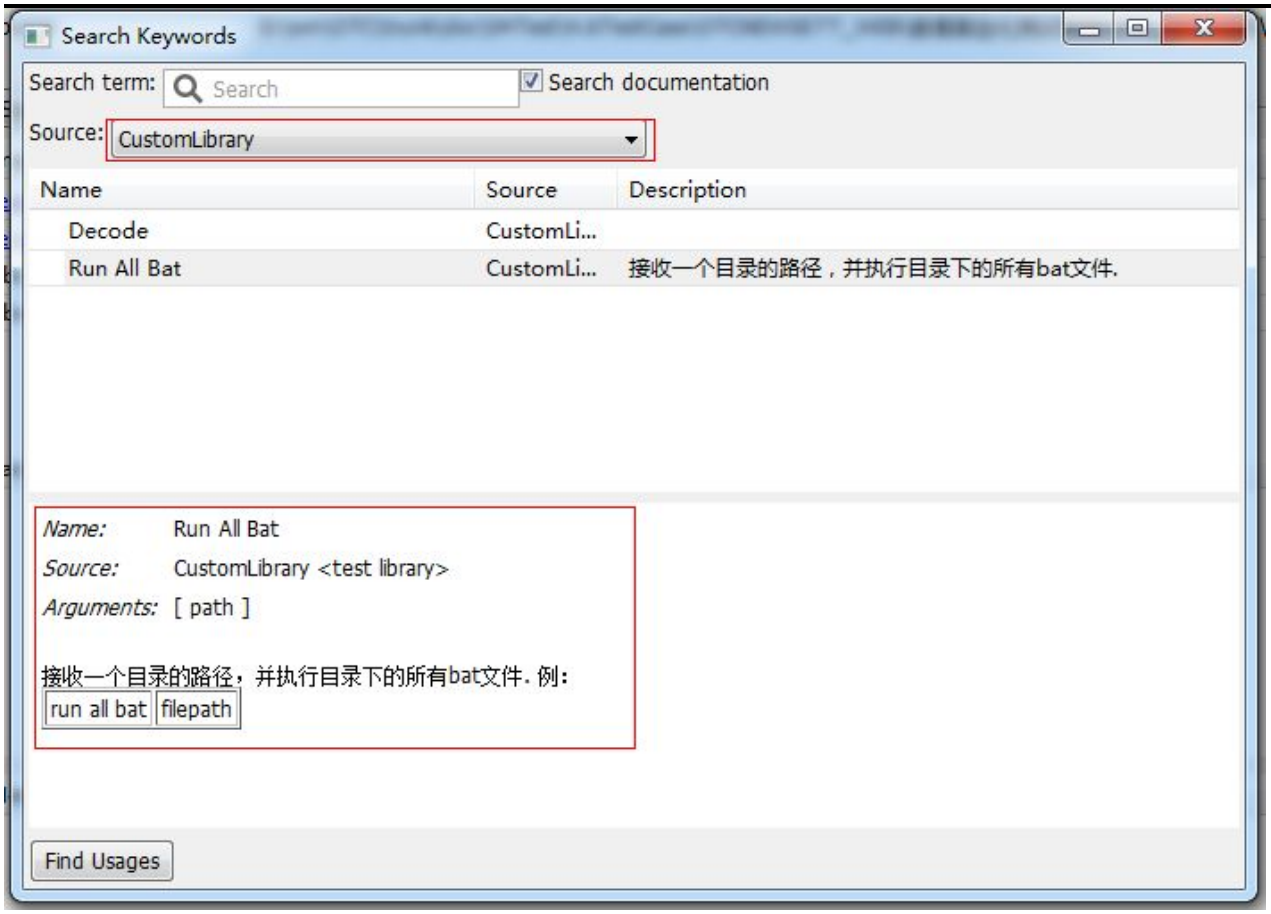
对于创建普通的模块来说这样已经 ok 了。但要想在 robot framework 启动后加载这个关键字，还需要在 CustomLibrary 目录下创建__init__.py 文件，并且它不是空的。

__init__py

```
from runbat import Runbat

__version__ = '0.1'
class CustomLibrary(Runbat):
    """
    这里也可以装 x 的写上我们创建的 CustomLibrary 如何如何。
    """

    ROBOT_LIBRARY_SCOPE = 'GLOBAL'
```

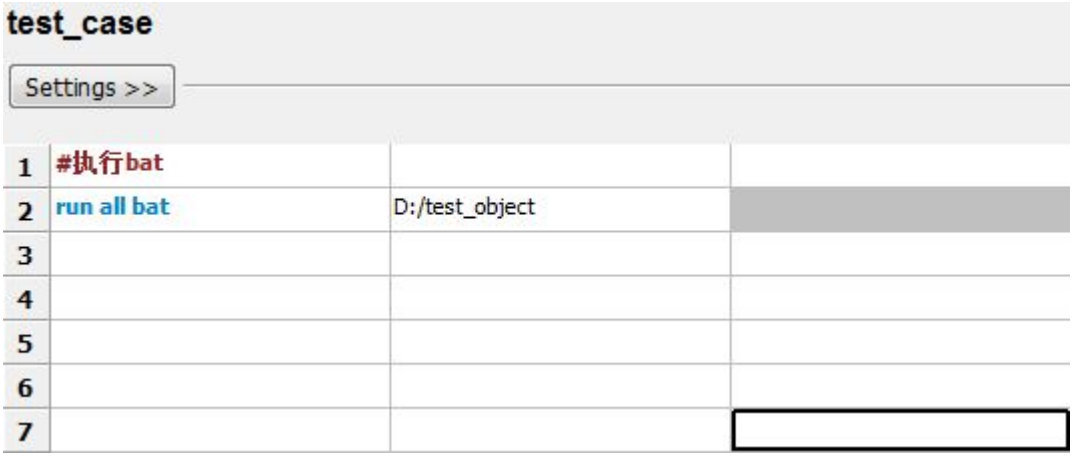
这个文件中其实有用的信息就三行，但必不可少。robot framwork 在启动时会加载这个文件，因为在这个文件里指明了有个 runbat 文件下面有个 Runbat 类。从而加载类里的方法（run_all_bat()）。

下面，启动 RIDE，按 F5:



找到了我们创建的关键字，下面就是在具体的项目或测试套件中引用 CustomLibrary

然后，在具体的测试用例中使用“run all bat” 关键字。



其实核心还是会点 Python ，利用工具，但又不受制于工具。