（一）Robot Framework 介绍

发布时间 2017 年 9 月 28 日 虫师

Robot Framework 官方网站：http://robotframework.org/

介绍

Robot Framework 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。

```
test case

open  browser       http://www.baidu.com      chrome

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-seleniumlibrary** 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）**。

特点

Clear

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

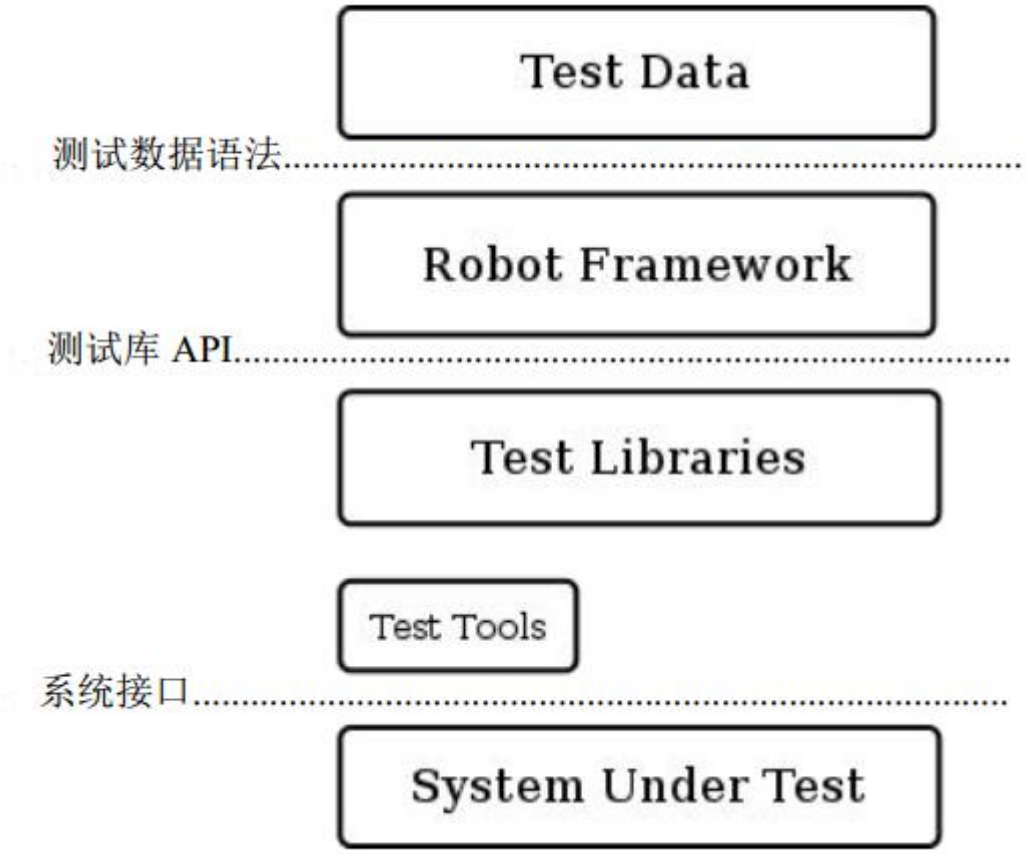
Test execution is started from the command line. As a result you get report and log in HTML format as well as an XML output. These provide extensive look into what your system does.

以上是官网的介绍，我懒得翻译了。我将 Robot Framework 的特点总结为以下几点：

- 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

模块化 & 架构

Modular



Robot framework 本质上是基于 **Python** 语言开发的一个框架，它提供了一套独立的语法。它本身只提供基础的一些功能。比如，它自带的 **Builtin** 库中提供的关键字告诉你如何定义变量、数组和字典，打印信息，分支语句和循环等。以及框架本身所提供的“自动化” 功能，如何组织用例，生成测试报告。

如果你想实现某一类型的自动化测试，如中接口、**UI** 或 移动 **APP** 的自动化，需要通过第三方 **Library** 完成。

支持的 Library

Robot Framework 所支持的库主要分 标准库 、 扩展库 和 其它 。 标准库提供基本功能，扩展库提供特定领域的操作。

因为 **Robot Framework** 所支持的测试库非常多，这里例一些常用的。

- **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。
- 查看所有 Librarys

例子

最后我们一睹 Robot Framework 语法的风采。

```
*** Settings ***
Documentation     A test suite with a single test for
 valid login.
...
...               This test has a workflow that is cr
eated using keywords in
...               the imported resource file.
Resource          resource.txt

*** Test Cases ***
Valid Login
    Open Browser To Login Page
    Input Username    demo
    Input Password    mode
    Submit Credentials
    Welcome Page Should Be Open
    [Teardown]    Close Browser
```

（二）Robot Framework 安装

发布时间 2017 年 9 月 28 日 虫师

安装 Robot Framework

如果想使用 **Robot Framework** 必须要安装：

-

Python 编程语言，参考 。

-
- Robot Framework
- Robot Framework 推荐 pip 方法安装（在 Windows 命令提示符（cmd）/ Linux 终端输入）：

```
λ pip install robotframework

Collecting robotframework
  Using cached robotframework-3.0.2.tar.gz
Installing collected packages: robotframework
  Running setup.py install for robotframework ... done
Successfully installed robotframework-3.0.2
```

安装 RIDE（可选）

如果你使用的是 Python2.x 版本，虽然 Python2.x 预计到 2020 年停止维护了，但仍然不少人在使用，Robot Framework 的所有相关库也没有完全迁移到 Python3.x。

比如 Robot Framework RIDE ，它是编写 Robot Framework 的标准编辑器。对于新手来降低了 Robot Framework 的使用门槛。

接下来安装 RIDE（只针对 Python2.x 环境）

- 安装 wxPython

下载地址: http://sourceforge.net/projects/wxpython/files/wxPython/2.8.12.1/

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。必须是 wxPython 2.8.12.1 版本，RIDE 基于该版本开发。

- 安装 RIDE

推荐 pip 安装

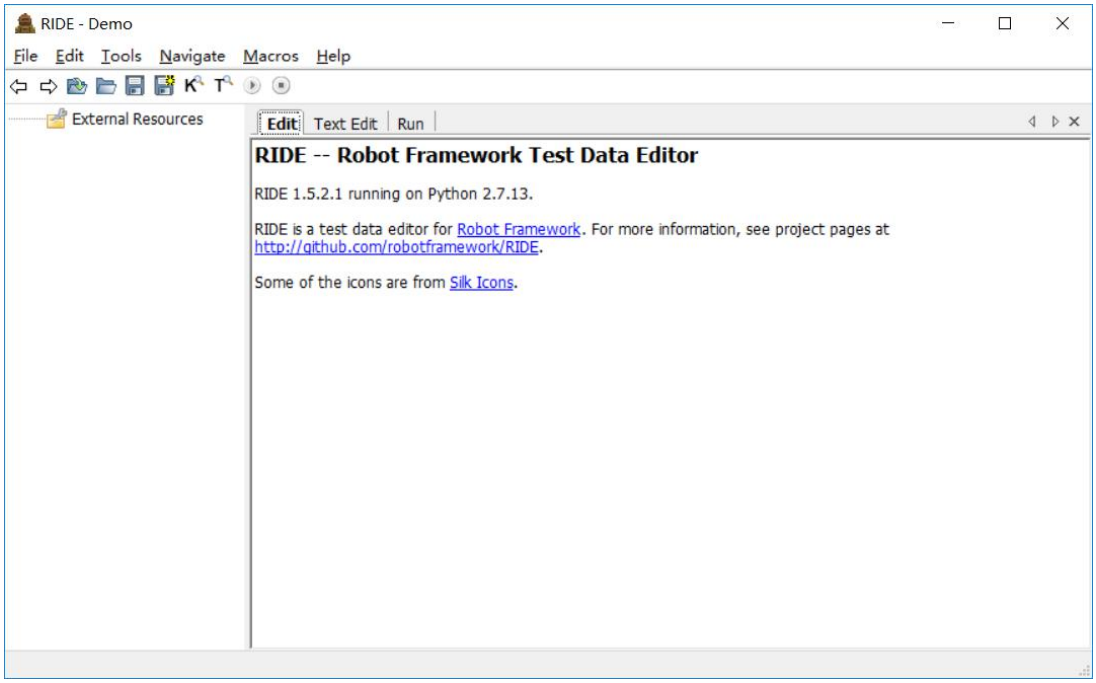
```
λ pip install robotframework-ride



Collecting robotframework-ride

  Using cached robotframework-ride-1.5.2.1.tar.gz

Installing collected packages: robotframework-ride

  Running setup.py install for robotframework-ride ... done

Successfully installed robotframework-ride-1.5.2.1
```

RIDE 是 Robot Framework 的官方编辑器。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。

- 启动 RIDE

切换到 Python2.7.x 的 Script 目录（例如：C:\Python27\Scripts），运行 ride.py 文件。

```
C:\Python27\Scripts> python ride.py
```



Python3.x 可用的编辑器

那身为一个 **Python3.x** 的用户应该使用什么编辑器来开发 **Robot Framework** 呢？

Robot Framework 提供了主流编辑器的插件，这里。 **Atom、Eclipse、Notepad++、IntelliJ IDEA、Sublime text 、Vim** 都可以找到对应的插件。

这里选择 **Sublime text3** 为例。

- 安装 Sublime text3

官网：**http://www.sublimetext.com/**

- 下载（克隆）sublime-robot-framework-assistant

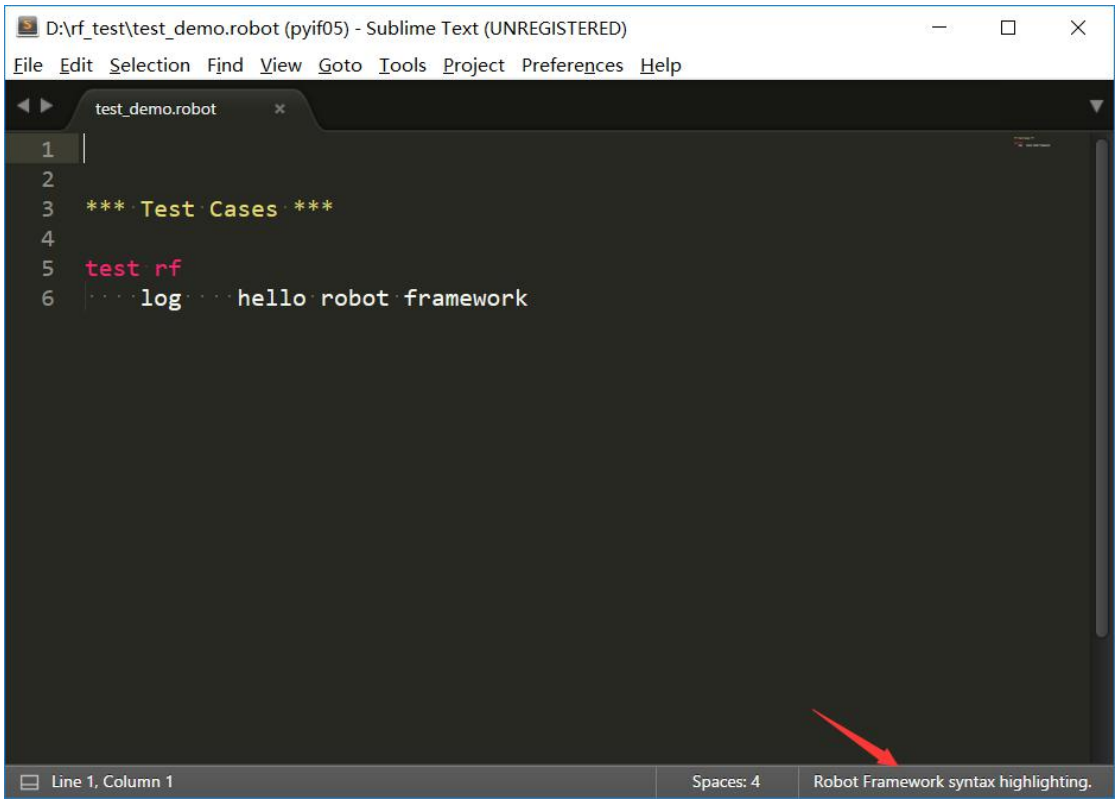
项目地址：https://github.com/andriyko/sublime-robot-framework-assistant

将整个 sublime-robot-framework-assistant-master 项目目录放拷贝到 Sublime Text3 的 Packages 目录下，然后，重启动 Sublime Text3。

(打开 sublime text 编辑器，菜单栏：Perferences -> Browser Packages... 打开的目录就是 Packages 目录了!)

在 Sublime Text3 菜单栏选择 “View” ->“Syntax” -> “Robot Framework syntax highlighting” ，选择 Robot Framework 语法高亮。

最终效果如下：



（三）Robot Framework 创建测试、运行与生成报告

发布时间 2017 年 9 月 28 日 虫师

上一节我们已经介绍 Robot Framework-RIDE 只支持 Python2 ，但 Python2 到 2020 年将不再维护，所以接下来的关于 Robot Framework 的学习将不再基于 Robot Framework-RIDE，你可以参考上一节中介绍的 Sublime Text3 + sublime-robot-framework-assistant 插件来编写 Robot Framework 脚本。

创建测试

- 测试项目（目录）： rf_test/
-
- 测试套件（文件）： test_suit.robot
-
- 测试用例 （test_suit.robot 文件中代码）：

```
*** Test Cases ***


test case1
```

```
log      hello  robot  framework
```

运行测试

Robot Framework 运行测试通过 pybot 命令，检查 _C:\Python36\Scripts_ 目录下是否有 pybot.bat 文件，正确安装 Robot Framework 一定会生成该文件。

C:\Python36\Scripts 目录一定要添加环境变量 path。

打开 cmd Window 命令提示符，切换到 Robot Framework 项目目录。

- 运行一条用例：

```
...\rf_test>  pybot  --test  test_case  test_suit.robot
```

- 运行指定文件：

```
...\rf_test>  pybot  test_suit.robot
```

- 运行当前目录下以.robot 为后缀名的测试文件

```
...\rf_test>  pybot  *.robot
```

- 运行当前 test_a 目录下的所有用例

```
...\rf_test>  pybot  test_a
```

- 运行当前目录下的所有以.robot 为后缀名的测试文件

```
...\rf_test> pybot ./
```

生成测试报告

当通过上面的命令运行测试，**Robot Framework** 会自动帮我们生成测试报告。

```
D:\rf_test > pybot test_suit.robot

==============================================================================
==============
Test Suit
==============================================================================
==============

test case1
| PASS |

------------------------------------------------------------------------------
------------------
Test Suit
| PASS |

1 critical test, 1 passed, 0 failed

1 test total, 1 passed, 0 failed
```

```
==============================================================================
==============

Output:  D:\rf_test\output.xml

Log:     D:\rf_test\log.html

Report:  D:\rf_test\report.html
```

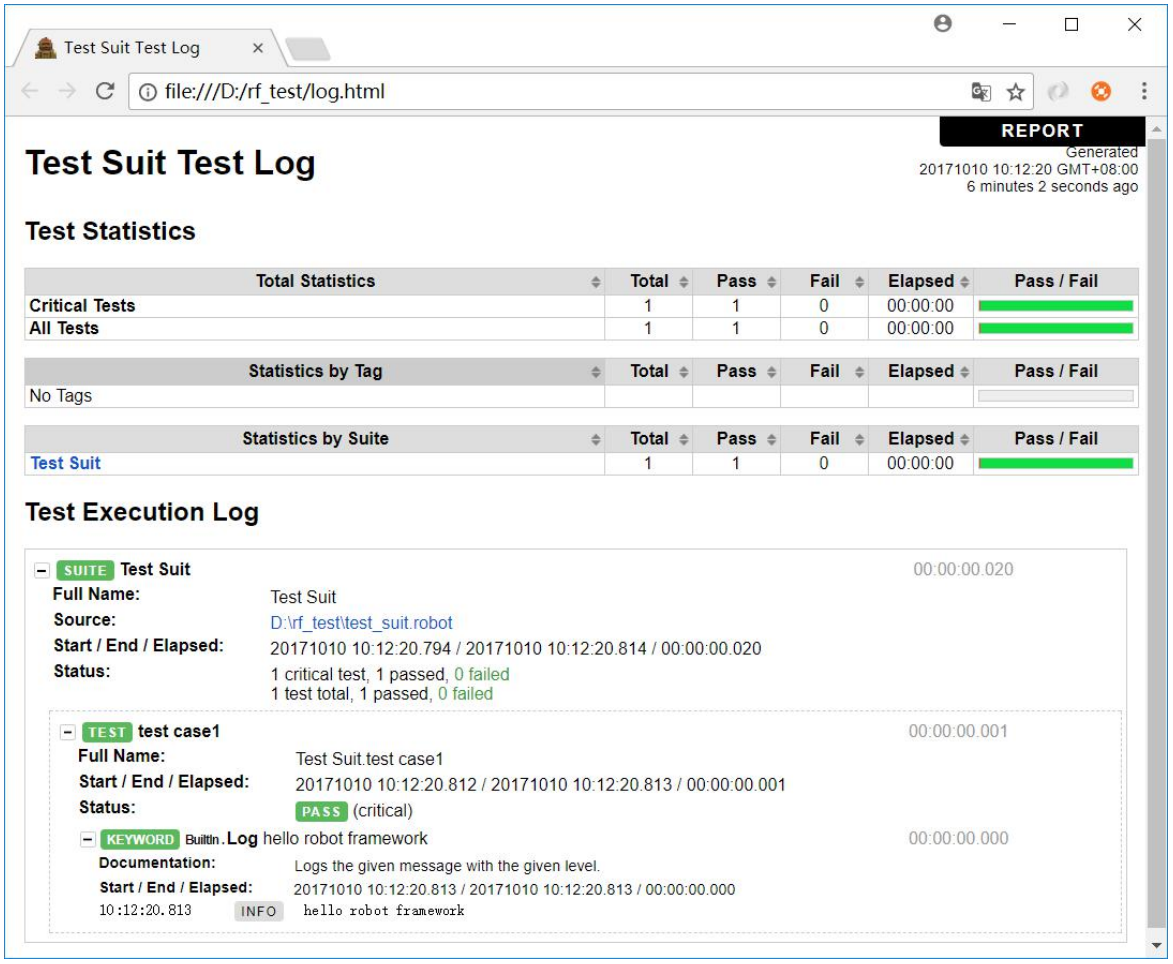
当用例运行结束，**Robot Framework** 生成三个文件：**output.xml**、**log.html** 和 **report.html**。

output.xml 记录的测试结果是 **XML** 文件。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。

log.html 会记录 **Robot Framework** 运行的每一步操作，主要用于编写测试脚本的过程中查看。

report.html 为测试报告，整理性的展示测试用例的运行情况。

通过浏览器打 **log.html** 文件查看。



怎么样，相信通过这一节的学习，你已经学会了 **Robot Framework** 基本使用。

（四）Robot Framework 基础关键字

发布时间 2017 年 9 月 28 日 虫师

相信你已经迫不及待的要拿 **Robot Framework** 写自动化测试项目了，先别着急！ 当你要使用 **Python** 去开发一个网站的时候，是不是要先从 **Python** 的基本语法学起？ **Builtin** 库是 **Robot Framework** 自带的基础库，提供了一套基础的关键字。本节介绍的大多关键字都由该库提供。

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。

```
*** Test Cases ***

test case1
    log    robot framework
    log    python
```

定义变量

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

```
*** Test Cases ***

test case2
    ${a}    Set variable    python
```

```
    log    ${a}
```

连接对象

“Catenate”关键字可以连接多个对象

```
*** Test Cases ***


test case3
    ${hi}    Catenate    hello    world
    log    ${hi}
```

加上**“SEPARATOR=”** 可以对多个连接的信息进行分割。

```
*** Test Cases ***
```

```
test  case4
    ${hi}    Catenate    SEPARATOR=---    hello    world
    log    ${hi}
```

定义列表

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

```
*** Test Cases ***

test  case5
    @{abc}    Create List    a    b    c
    log many    @{abc}
```

时间操作

在 **Robot Framework** 中也提供操作时间的关键字。

1、 “get time” 关键字用来获取当前时间。

```
*** Test Cases ***

test case6
    ${t}    get time
    log    ${t}
```

2、 “sleep” 关键字用来设置休眠一定时间

```
*** Test Cases ***

test case7
```

```
    ${t}      get  time

    sleep      5

    ${t}      get  time
```

（五）Robot Framework 基础关键字（高级用法）

发布时间 2017 年 9 月 28 日 虫师

上一小节，你已经感受到了 **Robot Framework** 的基本说法，这一小节你将会看到 **Robot Framework** 更多强大的用法。

if 语句

通过 **“run keyword if”** 关键字可以编写 **if** 分支语句。

```
*** Test Cases ***



test case8

    ${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”。

注意 sele if 和 else 前面的三个点点点（...）

for 循环

在 Robot Framework 中编写循环通过 “:FOR” 。

1、循环 0~9 。

```
*** Test Cases ***
```

```
test case9
    :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。

2、遍历列表。

```
*** Test Cases ***


test case10
    @{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。

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。

1、 生成随即数

在 Python 中我们可以这样来引和并使用方法：

```
C:\Users\fnngj> python



Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64
bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import random

>>> random.randint(1000, 9999)

9777
```

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。

```
*** Test Cases ***

test case11
    ${d}    Evaluate    random.randint(1000, 9999)    random
    log    ${d}
```

2、 调用自己编写的 Python 程序

首先创建 D:/rf_test/count.py 文件。

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

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

通过 add 函数实现两个数相加，这是再简单不过的小程序了。

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

```
*** Test Cases ***

test case12
    Import Library    D:/rf_test/count.py
    ${a}    Evaluate    int(4)
    ${b}    Evaluate    int(5)
    ${add}    add    ${a}    ${b}
    log    ${add}
```

```

```

在 Robot Framework 中所有的内容都不是字符串类型，所以，需要通过“Evaluate” 将 4 和 5 转化为 int 类型后，再调用 add 计算两个数的和。

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。除此之外，你也可以像 Python 一样使用“#”号进行注释。

```
*** Test Cases ***


test case13
    Comment    这是注释
    #这也是注释
```

如果你熟悉 Python 编程语言，那么 Robot Framework 几乎没有实现不了的功能。

（六）扩展库之 SeleniumLibrary 安装与运行

发布时间 2017 年 9 月 28 日 虫师

最算学到 Selenium 了，希望你没被前面的 Robot Framework 基本语法吓退！

SeleniumLibrary 是针对 Robot Framework 开发的 Selenium 库。它也 Robot Framework 下面最流程的库之一。主要用于编写 Web UI 自动化测试。

安装 SeleniumLibrary

项目地址：**https://github.com/robotframework/SeleniumLibrary**

SeleniumLibrary（推荐使用 pip 命令）安装：

```
> pip install --pre --upgrade robotframework-seleniumlibrary



Collecting robotframework-seleniumlibrary

  Downloading
robotframework_seleniumlibrary-3.0.0b3-py2.py3-none-any.whl (72kB)

    100%
|████████████████████████████████████████|
81kB 44kB/s

Requirement already up-to-date: robotframework>=2.8.7 in
c:\python36\lib\site-packages (from robotframework-seleniumlibrary)

Requirement already up-to-date: selenium>=2.53.6 in
c:\python36\lib\site-packages (from robotframework-seleniumlibrary)
```

```
Installing collected packages: robotframework-seleniumlibrary

Successfully installed robotframework-seleniumlibrary-3.0.0b3
```

编写第一个例子

创建 **robot_se.robot** 文件。调用 **SeleniumLibrary** 中所提供的关键字，编写 **Web** 自动化测试。

```
*** Settings ***

Library                 SeleniumLibrary



*** Test Cases ***

Baidu search case

    Open Browser    http://www.baidu.com    chrome

    Input text    id=kw    robot framework

    click button    id=su

    close Browser
```

代码解析：

*** Settings ***： 用来定义设置部分。

Library SeleniumLibrary：在 Robot Framework 脚本中导入 SeleniumLibrary 模块

Baidu search case： 创建一条百度搜索的测试用例。

Open Browser： SeleniumLibrary 关键字，用于启动浏览器，并指定打开网址。

Input text： SeleniumLibrary 关键字，用于向输入框中输入内容。id=kw 定位要操作的元素。

click button： SeleniumLibrary 关键字，用于点击元素。id=su 定位要操作元素

close Browser： SeleniumLibrary 关键字，用于关闭浏览器。

运行测试

通过 pybot 命令运行测试。

```
λ  pybot  robot_se.robot



==============================================================================
==============
Robot Se :: Simple example using SeleniumLibrary.
==============================================================================
==============
```

```
Baidu search case
| PASS |

------------------------------------------------------------------------------

Robot Se :: Simple example using SeleniumLibrary.                             |
PASS |

1 critical test, 1 passed, 0 failed

1 test total, 1 passed, 0 failed

==============================================================================

Output:  D:\rf_test\robotSe\output.xml

Log:     D:\rf_test\robotSe\log.html

Report:  D:\rf_test\robotSe\report.html
```

运行结果你可以通过生成的 **log.html** 或 **report.html** 文件查看。

selenium3 浏览器驱动

发布时间 2017 年 6 月 29 日

下载浏览器驱动

当 **selenium** 升级到 **3.0** 之后，对不同的浏览器驱动进行了规范。如果想使用 **selenium** 驱动不同的浏览器，必须单独下载并设置不同的浏览器驱动。

设置浏览器驱动

设置浏览器的地址非常简单。我们可以手动创建一个存放浏览器驱动的目录，如：**C:\driver**，将下载的浏览器驱动文件（例如：**chromedriver**、**geckodriver**）丢到该目录下。

我的电脑->属性->系统设置->高级->环境变量->系统变量->Path，将“C:\driver”目录添加到 **Path** 的值中。

- Path
- ;C:\driver

设置浏览器驱动

验证不同的浏览器驱动是否正常使用。

```
from selenium import webdriver




driver = webdriver.Firefox()    # Firefox 浏览器
```

```
driver = webdriver.Chrome()      # Chrome 浏览器

driver = webdriver.Ie()          # Internet Explorer 浏览器

driver = webdriver.Edge()        # Edge 浏览器

driver = webdriver.Opera()       # Opera 浏览器

driver = webdriver.PhantomJS()   # PhantomJS

.....
```

（七）扩展库之 SeleniumLibrary 元素定位

发布时间 2017 年 9 月 28 日 虫师

SeleniumLibrary 元素定位

SeleniumLibrary 提供了两种指定前缀的显式定位器策略。第一种：

```
strategy:value
```

这种语法只支持 SeleniumLibrary 3 版本以上，是新的定位写法。

第二种：

```
strategy=value
```

这种语法是 Robot Framework 通常所使用的命令参数的语法。

SeleniumLibrary 支持的元素方法:

= Strategy =	= Match based on =	= Example =
id	Element id.	id:example
name	name attribute.	name:example

= Strategy =	= Match based on =	= Example =
identifier	Either id or name.	identifier:example
class	Element class.	class:example
tag	Tag name.	tag:div
xpath	XPath expression.	xpath://div[@id="example"]
css	CSS selector.	css:div#example
dom	DOM expression.	dom:document.images[5]
link	Exact text a link has.	link:The example
partial link	Partial link text.	partial link:he ex
sizzle	Sizzle selector provided by jQuery.	sizzle:div.example
jquery	Same as the above.	jquery:div.example
default	Keyword specific default behavior.	default:example

分割符号的空格将被忽略，所以， **id : foo**，**id: foo** 和 **id:foo** 都是等价的。

例如：

```
demo                    –
Click  Element          id:container
Click  Element          css:div#container  h1
Click  Element          xpath:  //div[@id=“container”  ]//h1
```

如果定位器的开头为“//” 或“(//” 测被当做 **Xpath** 定位。换句话说，用 **//div** 和 **xpath://div** 是等价的。

例如：

```
demo                    –
Click  Element          //div[@id=“container”  ]
Click  Element          (//div)[2]
```

除了可以直接操作元素外，也可以通过 **Get WebElement** 关键字获取元素对象。

```
demo                    –                       –
${elem}  =              Get  WebElement         id=example
Click  Element          ${elem}
```

如果想更详细的了解元素定位的用法，可以阅读 **SeleniumLibrary** 库中的 **__init__.py** 文件

（八）扩展库之 SeleniumLibrary 常用关键字

发布时间 2017 年 9 月 28 日 虫师

1. 浏览器驱动

通过不同的浏览器执行脚本

```
Open  Browser      htpp://www.testclass.net      chrome
```

浏览器对应的关键字：

关键字	浏览器/设备
firefox	FireFox
ff	FireFox
internetexplorer	Internet Explorer
ie	Internet Explorer
googlechrome	Google Chrome
gc	Google Chrome
chrome	Google Chrome
opera	Opera
phantomjs	PhantomJS
htmlunit	HTMLUnit

关键字	浏览器/设备
htmlunitwithjs	HTMLUnit with Javascipt support
android	Android
iphone	Iphone
safari	Safari
edge	Edge

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。
-
- 如果不设置浏览器，默认打开 Firefox.
-

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。

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。

4. 设置浏览器窗口宽、高

```
Get Window Size    800    600
```

get window 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 window 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. 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。

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。

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。

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”

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。

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。

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。@name 获取这个元素的 name 属性值。

12. cookie 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 cookie （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。

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”。

获得文本信息进行比较

```
${text}      Get  Text

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

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。

15. 下拉框选择

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

Unselect From List By Value 关键字用于选择下拉框。 **Xpath=//*[@]** 定位下拉框；**Vlaue** 选择下拉框里的属性值。

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

了解了 **SeleniumLibrary** 所提供的这些关键字后你就可以开始动手写自动化测试了。

（九）扩展库之 SeleniumLibrary Web 测试

发布时间 2017 年 9 月 28 日 虫师

上一小节介绍介绍了 **SeleniumLibrary** 的常用关键字，这一节来举两个例子。

百度搜索实例

```
*** Settings ***

Documentation       Simple example using SeleniumLibrary.

Library             SeleniumLibrary
```

```
*** Test Cases ***
Baidu search case
    Open Browser    https://www.baidu.com    chrome
    Input text    id:kw    selenium
    click button    id:su
    Evaluate    time.sleep(2)    time
    ${title}    Get Title
    should contain    ${title}    selenium_百度搜索
    close Browser
```

其中 **sleep** 为 **Python** 所提供的休眠方法。**Get Title** 获得搜索之后的页面标题，通过 **should contain** 关键字来断言标题是否正确。

126 邮箱登录实例

```
*** Settings ***
Documentation    Simple example using SeleniumLibrary.
Library          SeleniumLibrary
```

```
*** Test Cases ***
Mial login case
    Open Browser    http://www.126.com    chrome
    Evaluate    time.sleep(3)    time
    Select Frame    xpath=//*[@id="x-URS-iframe"]
    Input text    name:email    username
    Input text    name:password    123456
    click element    id:dologin
    Unselect Frame
    close Browser
```

这里主要使用了 **Select Frame** 关键字切换表单，登录按钮要用 **click element** 关键字。

（十）扩展库之 SeleniumLibrary 分层测试

发布时间 2017 年 9 月 28 日 虫师

这一节来介绍分层的概念，在编写自动化测试时经常会遇到重复的操作，分层的概念就是把重复的操作封装成“用户关键字”，这样就可以减少冗余。

百度搜索实例

同样以百度搜索为例，当我们多个用例都是使用百度搜索，只是每次输入的关键字不一样，那么就可以对百度的搜索操作进行封装。

```
*** Settings ***

Documentation       Simple example using SeleniumLibrary.

Library             SeleniumLibrary




*** Variables ***

${URL}              https://www.baidu.com

${BROWSER}          Chrome




*** Test Cases ***

case1

    Open Browser    ${URL}    ${BROWSER}
```

```
    ${title}    Baidu  Search    robot  framework

    should  contain    ${title}    robot  framework_百度搜索

    close  browser


case2

    Open  Browser    ${URL}    ${BROWSER}

    ${title}    Baidu  Search    selenium

    should  contain    ${title}    selenium_百度搜索

    close  browser




*** Keywords ***
Baidu  Search

    [Arguments]    ${search_key}

    Input  text    id:kw    ${search_key}

    click  button    id:su
```

```
    Evaluate      time.sleep(2)      time

    ${title}      Get  Title

    [Return]      ${title}
```

*** **Variables** *** 用于定义公共变量。**${URL}** 和 **${BROWSER}** 为定义的公共变量，

*** **Keywords** *** 用于定义用户关键字，**Baidu Search** 为关键字的名称，

[Arguments] 定义入参， **[Return]** 定义出参。

最后，分别在 **case1** 和 **case2** 中调用 **Baidu Search** 关键字。从而简化了测试用例本身，它只关注搜索的关键字和结果断言。

要想理解这个例子，你需要有一定编程的思想，理解类方法的调用。

十一）用 Python 写 Robot Framework 测试

发布时间 2017 年 9 月 28 日 虫师

我前面已经介绍过 **Robot Framework** 框架是基于 **Python** 语言开发的，所以，它本质上是 **Python** 的一个库。

这一节写给：

1、你懂 **Python** 语言。

2、又想使用 **Robot Framework** 测试框架，因为它提供了很好的测试报告。

百度搜索实例

创建 **py_robot.py** 文件，代码如下：

```
from robot.api import TestSuite

from robot.api import ResultWriter

from robot.model import Keyword




# 百度搜索测试

class BaiduSearchTest:



    def __init__(self, name, librarys=["SeleniumLibrary"]):

        # 创建测试套件

        self.suite = TestSuite(name)



        # 导入 SeleniumLibrary
```

```
        for lib in librarys:
            self.suite.resource.imports.library(lib)


    # 定义变量
    def create_variables(self):
        variables = {
            "${baidu}": "https://www.baidu.com",
            "${browser}": "Chrome",
            "${search_input}": "id=kw",
            "${search_btn}": "id=su"

        }
        for k, v in variables.items():
            self.suite.resource.variables.create(k, v)
```

```
    # 测试用例：启动浏览器

    def open_browsers(self):

        test_01 = self.suite.tests.create("启动浏览器")

        test_01.keywords.create("Open Browser",

            args=["${baidu}", "${browser}"])

        test_01.keywords.create("Title Should Be",

            args=["百度一下，你就知道"])



    # 测试用例：百度搜索测试

    def search_word(self):

        test_02 = self.suite.tests.create("百度搜索测试")

        test_02.keywords.create("Input Text",

            args=["${search_input}", "测试教程网"])

        test_02.keywords.create("Click Button",

            args=["${search_btn}"])
```

```
        test_02.keywords.create("Sleep",  args=["5s"])



    # 测试用例：断言验证搜索结果标题

    def  assert_title(self):

        test_03  =  self.suite.tests.create("断言验证搜索结果标题")

        test_03.keywords.create("Title  Should  Be",

            args=["测试教程网_百度搜索"])



    # 测试用例：关闭测试用例

    def  close_browsers(self):

        test_04  =  self.suite.tests.create("关闭浏览器")

        test_04.keywords.create("Close  All  Browsers")



    # 运行

    def  run(self):
```

```
        self.create_variables()

        self.open_browsers()

        self.search_word()

        self.assert_title()

        self.close_browsers()



        # 运行套件

        result = self.suite.run(critical="百度搜索",

            output="output.xml")



        # 生成日志、报告文件

        ResultWriter(result).write_results(

            report="report.html", log="log.html")



if __name__ == "__main__":
```

```
    print("用 Python 写 Robot Framework 测试")

    suite = BaiduSearchTest("百度搜索测试套件")

    suite.run()
```

这段代码的运行通过 **python** 命令来执行。

```
> python py_robot.py



用 Python 写 Robot Framework 测试

==============================================================================

==============

百度搜索测试套件

==============================================================================

==============

启动浏览器

DevTools listening on

ws://127.0.0.1:12950/devtools/browser/bcbf14bb-ebc4-425c-882f-445

31afd9689

启动浏览器

| PASS |
```

```
------------------------------------------------------------------------------
百度搜索测试                                                          | PASS |
------------------------------------------------------------------------------
断言验证搜索结果标题                                                  | PASS |
------------------------------------------------------------------------------
关闭浏览器                                                            | PASS |
------------------------------------------------------------------------------
百度搜索测试套件                                                      | PASS |

0 critical tests, 0 passed, 0 failed

4 tests total, 4 passed, 0 failed
```

```
==============================================================================
==============

Output:  D:\rf_test\robotSe\output.xml
```

Robot Framework 用的好，Python 少不了！所以，我的建议是要想用好 Robot Framework 必须要学习和掌握 Python 语言。