

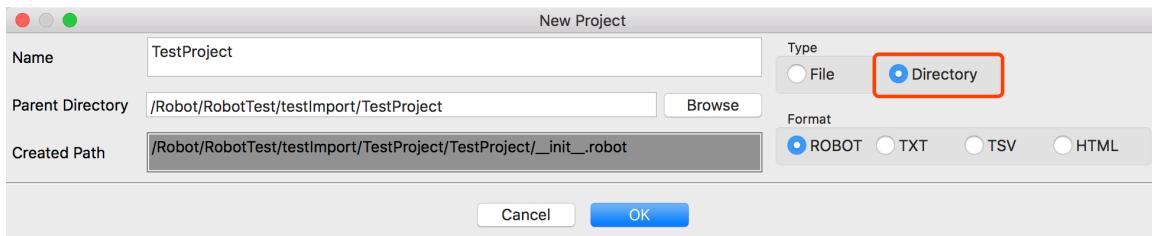
# 一. Robot 使用

## 1. 创建测试项目

菜单栏: File – New Project。

Name 填写项目名称。

Type 类型选择 Directory。

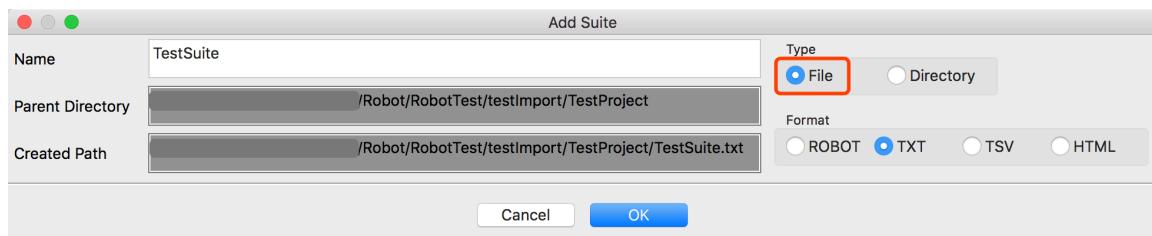


## 2. 创建测试套件

右键点击上一步创建的测试项目“TestProject” – 点击“New Suite”。

Name 填写测试套件名称。

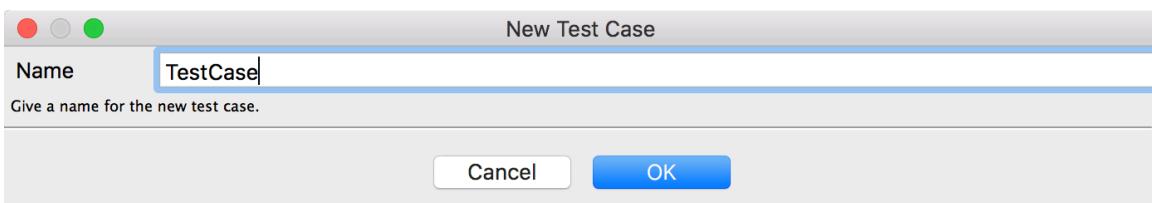
Type 类型选择 File。



## 3. 创建测试用例

右键点击上一步创建的测试套件“TestSuite” – 点击“New Test Case”。

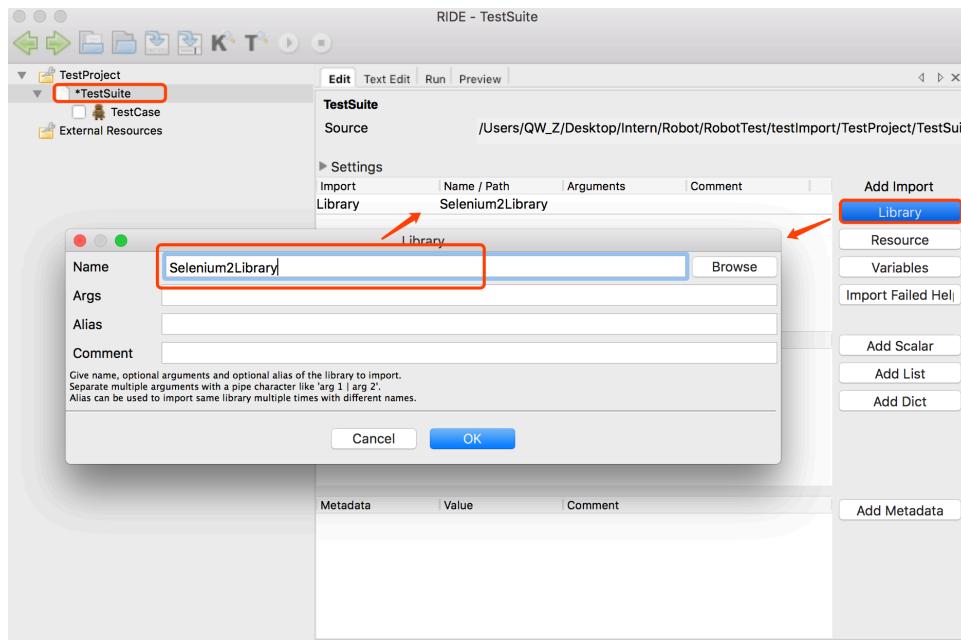
Name 填写测试用例名称。



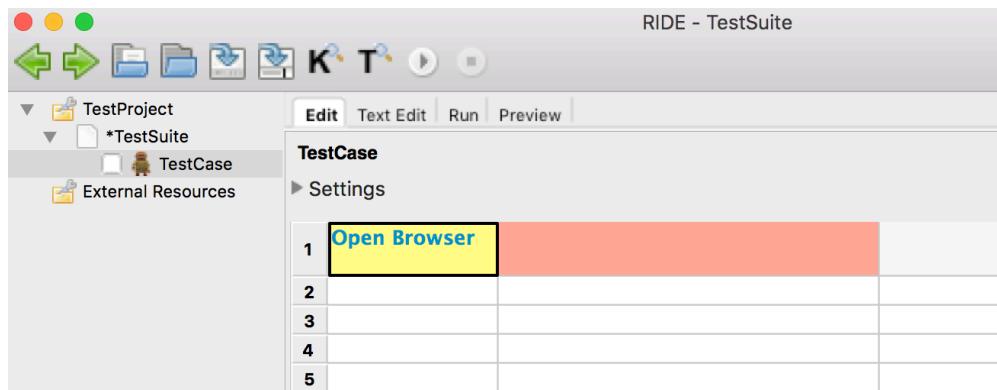
## 4. 导入库

在“TestSuite”的 Edit 标签页，点击“Library”按钮，弹出输入框，在 Name 处输入需要导入的包的名称，如 Selenium2Library，点击“OK”。

如果导入的库显示为黑色，表示导入成功；如果显示为红色，表示导入失败，库不存在。



## 5. 编写测试用例



“Open Browser”为关键字，变蓝说明此关键字合法。后面方框为红色，表示这个参数不能省。 (用#进行注释)

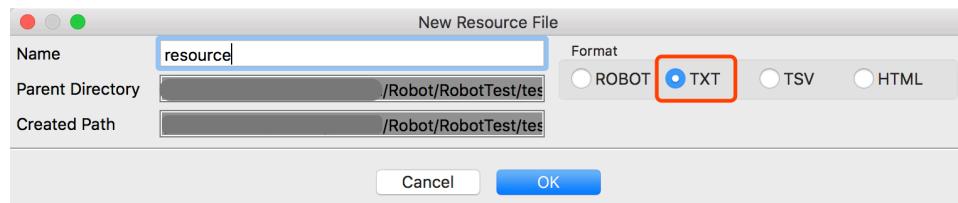
1	#在Safari中打开百度首页	百度主页网址	确定使用的浏览器
2	Open Browser	http:\\www.baidu.com\\	Safari

## 6. 创建并导入 Resource

a. 右键点击“TestProject” – 点击“New Resource”。

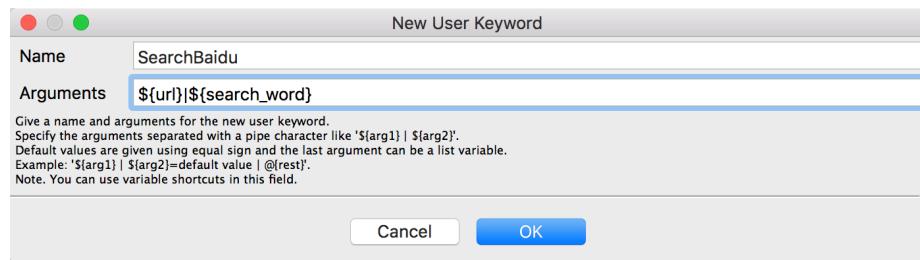
Name 填写 Resource 名称。

Format 默认为 TXT。

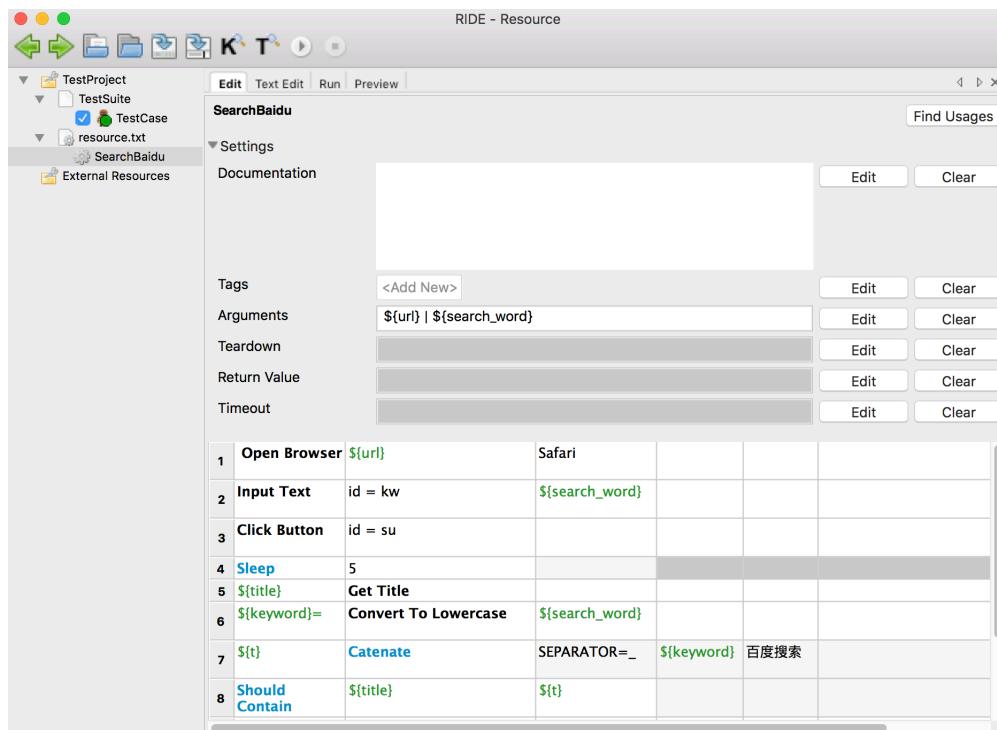


b. 右键点击“resource.txt” – 点击“New User Keyword”。

名字填写新建 Keyword 名称。如果新建的 Keyword 使用时需要输入参数，则在 Arguments 中输入需要的参数名称，以 | 分隔多个参数(参数也可以后来再添加)。



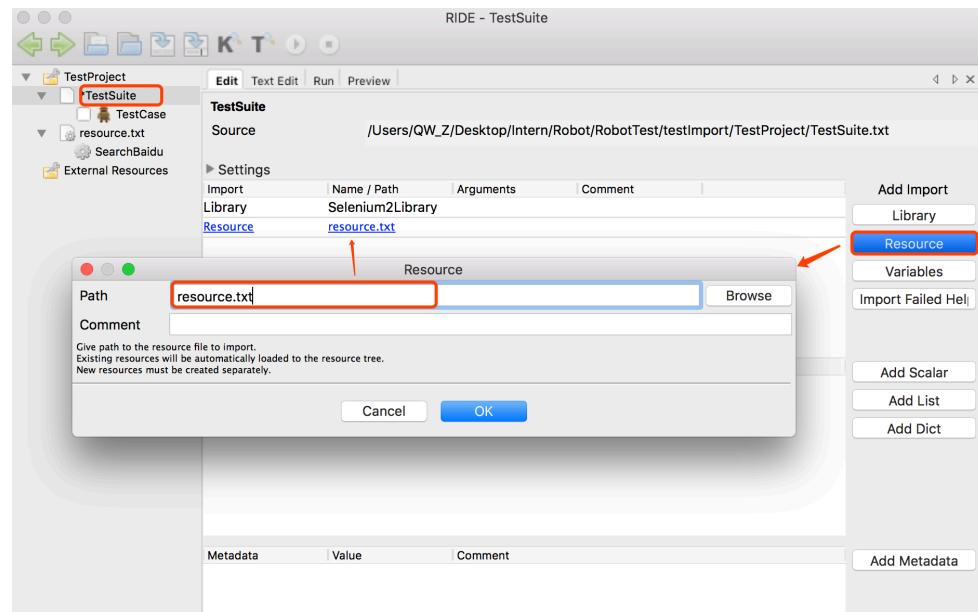
### c. 编写 Keyword 内容



New Keyword 中的关键字可以为黑色，但是引入此 Resource 文件的 TestSuite 中必须包含或引入此关键字依赖的包。

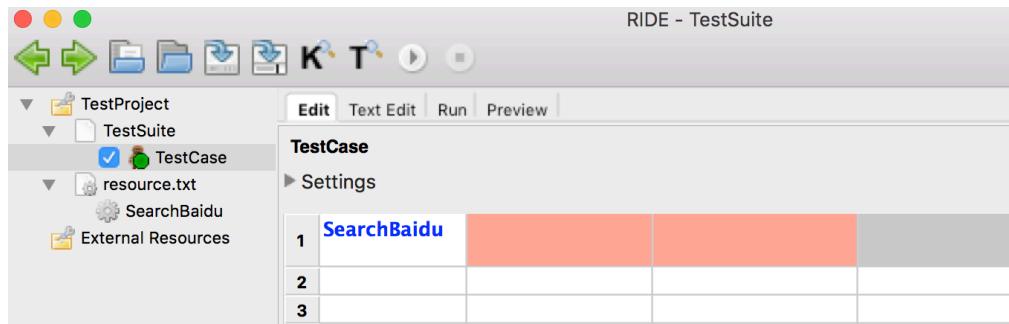
### d. 导入 Resource

点击“TestSuite” – 在 Edit 标签页，点击“Resource”。

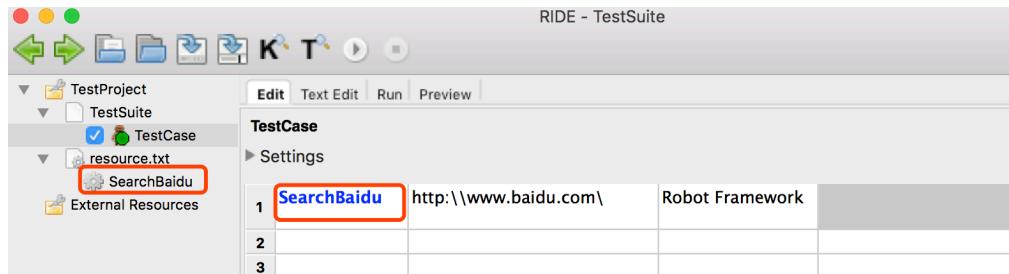


### e. 使用 Resource 中包含的 Keyword

在 TestCase 的“Edit”标签中编写测试用例内容。使用我们上一步定义在 Resource 中新关键字“SearchBaidu”。



两个红色的框表示我们刚才在关键字“SearchBaidu”内定义的两个参量 ( `${url} | ${search_word}`) 不能缺省。



注：每个 Test Suit 下用户新建的关键字只能被改 Test Suit 中的 Test Case 使用，但是 Resource 文件中包含的用户新建关键字可以被 Project 中任何一个导入了此 Resource 文件的 Test Suit 使用。

## 7. 定义 Setup 和 Teardown

Robot Framework 中的 Setup 和 Teardown 在各个级别的“Edit”标签页的 setting 中设置，并遵循以下几条原则。

- Suite 级别的 Setup\Teardown 是不会被覆盖的，但是 Test 级别的可以。
- 覆盖顺序是 Test case → Test Suite → Project/Directory。
- 当 TestCase 有自己的 Setup 或 Teardown 时，会选择实行自己的，抛弃上级的。
- 当 TestSuite 和 TestCase 都没有指定其 Setup\Teardown 时，就执行 Project/Directory。
- 当某个 TestCase 没有 Setup 或 Teardown，就执行上一级的；如果有的话，就执行自己对应的 Setup 或 Teardown。

## 二. Robot 常用关键字

### 1. Common FOR loop

- 在 IN 后给出所有变量

#Common Loop						
: FOR	<code> \${animal}</code>	IN	cat	dog	horse	
\	LOG		<code> \${animal}</code>			

`${animal}`代表每次循环从列表中取出的一个值，使用\作为改行的行首关键字。

输出结果：

```
Starting test: testImport.exMem.test2
20170629 10:11:05.933 : INFO : cat
20170629 10:11:05.938 : INFO : dog
20170629 10:11:05.941 : INFO : horse
```

## b. 从列表中赋值

<code> \${table}</code>	<code>set variable</code>	a	b	c	d
: FOR	<code> \${var}</code>	IN	<code>@{table}</code>		
\	<code>LOG</code>	<code> \${var}</code>			

首先定义一个列表\${table}并赋值，在FOR循环中，每次循环从定义的列表中取出一个值。\\后仍然接循环体。

输出结果：

```
20170629 10:17:44.251 : INFO : ${table} = [u'a', u'b', u'c', u'd', u'e']
20170629 10:17:44.253 : INFO : a
20170629 10:17:44.256 : INFO : b
20170629 10:17:44.258 : INFO : c
20170629 10:17:44.260 : INFO : d
20170629 10:17:44.261 : INFO : e
```

## 2. FOR Loop in Enumerate

当需要知道当前循环的循环位置(index)时，可以使用FOR index ... IN ENUMERATE ...关键字。固定使用\${index}作为index编号变量名。

: FOR	<code> \${index}</code>	<code> \${item}</code>	IN ENUMERATE	<code>@{table}</code>
\	<code>LOG</code>	<code> \${index} : \${item}</code>		

此循环体中，记录index值及当次循环所取的列表值

输出结果：

```
20170629 10:17:44.267 : INFO : 0 : a
20170629 10:17:44.269 : INFO : 1 : b
20170629 10:17:44.271 : INFO : 2 : c
20170629 10:17:44.273 : INFO : 3 : d
20170629 10:17:44.275 : INFO : 4 : e
```

## 3. 跳出循环

### a. 使用 Run Keyword If 关键字

“\${index} == 4”为判断的条件，“Exit For Loop”为符合条件时执行的关键字，即跳出循环。

若不符合条件则执行“LOG”关键字

: FOR	<code> \${index}</code>	<code> \${item}</code>	IN ENUMERATE	<code>@{table}</code>
\	<code>Run Keyword If</code>	<code> \${index}==4</code>	<code>Exit For Loop</code>	
\	<code>LOG</code>	<code> \${index} : \${item}</code>		

输出结果：

```
20170629 10:17:44.283 : INFO : 0 : a
20170629 10:17:44.288 : INFO : 1 : b
20170629 10:17:44.293 : INFO : 2 : c
20170629 10:17:44.299 : INFO : 3 : d
20170629 10:17:44.304 : INFO : Exiting for loop altogether.
```

### b. 使用 Exit For Loop If 关键字

: FOR	<code> \${index}</code>	<code> \${item}</code>	IN ENUMERATE	<code>@{table}</code>
\	<code>Exit For Loop If</code>	<code> \${index}==4</code>		
\	<code>LOG</code>	<code> \${index} : \${item}</code>		

当满足条件，即“\${index} == 4”时，跳出循环。输出结果同上。

## 4. Loop in Range

### a. 只使用数据上限

数据从0开始，每次+1，到指定数值，但不包含上限数值。

:FOR	<code> \${item}</code>	IN RANGE	10
\	<code>LOG</code>	<code> \${item}</code>	

输出结果:

```
20170629 10:41:13.957 : INFO : 0
20170629 10:41:13.960 : INFO : 1
20170629 10:41:13.963 : INFO : 2
20170629 10:41:13.965 : INFO : 3
20170629 10:41:13.966 : INFO : 4
20170629 10:41:13.968 : INFO : 5
20170629 10:41:13.970 : INFO : 6
20170629 10:41:13.973 : INFO : 7
20170629 10:41:13.977 : INFO : 8
20170629 10:41:13.980 : INFO : 9
```

b. 使用开始和结束数值

数据从指定的“开始数值”开始，每次+1，至“结束数值”，但不包含结束数值

:FOR	\${item}	IN RANGE	20	24	
\	LOG	\${item}			

输出结果:

```
20170629 10:41:13.984 : INFO : 20
20170629 10:41:13.986 : INFO : 21
20170629 10:41:13.989 : INFO : 22
20170629 10:41:13.991 : INFO : 23
```

c. 使用开始、结束和步长

数据从指定的“开始数值”开始，每次+“步长值”，至“结束数值”，但不包含结束数值。

:FOR	\${item}	IN RANGE	1	7	2
\	LOG	\${item}			

输出结果:

```
20170629 10:41:13.995 : INFO : 1
20170629 10:41:13.998 : INFO : 3
20170629 10:41:14.003 : INFO : 5
```

## 5. Repeat Keyword

#Repeat Keyword			
Repeat Keyword	3	LOG	Robot

“Repeat Keyword” + 重复次数 + 被重复执行的关键字

输出结果:

```
20170629 13:40:40.536 : INFO : Repeating keyword, round 1/5.
20170629 13:40:40.537 : INFO : Robot
20170629 13:40:40.538 : INFO : Repeating keyword, round 2/5.
20170629 13:40:40.540 : INFO : Robot
20170629 13:40:40.540 : INFO : Repeating keyword, round 3/5.
20170629 13:40:40.542 : INFO : Robot
20170629 13:40:40.542 : INFO : Repeating keyword, round 4/5.
20170629 13:40:40.545 : INFO : Robot
20170629 13:40:40.548 : INFO : Repeating keyword, round 5/5.
20170629 13:40:40.549 : INFO : Robot
```

## 6. LOG a List

LOG 关键字有很多参数，当 list 中的元素多于 1 个时，直接使用 LOG 会将 list 中的元素对应到每个参数中，会发生报错。所以，有以下两种 LOG list 的方法。

(其中 \${list\_copy} = {a, b, c, d, e})

a. 使用“Log Many”关键字

Log Many 后面可以直接接 List 类型。List 中的元素会依次被 LOG 出来

Log Many	@{list_copy}	
----------	--------------	--

输出结果:

```
20170629 13:51:35.025 : INFO : a
20170629 13:51:35.026 : INFO : b
20170629 13:51:35.026 : INFO : c
20170629 13:51:35.026 : INFO : d
20170629 13:51:35.027 : INFO : e
```

## b. 使用“Log”关键字

若直接使用 LOG 关键字，则需要将 list 型的变量强制转型为 Scalar，即将变量名前代表 list 型的@符号换为\$符号。这样输出的结果是一条字符串。

Log	\${list_copy}	
-----	---------------	--

输出结果：

```
20170629 13:51:35.028 : INFO : [u'a', u'b', u'c', u'd', u'e']
```

## 7. 获得 list 长度

也需要强制将 List 型转为 Scalar。

Get Length	\${list_copy}
------------	---------------

## 三. Robot 简单实例

用例实现在 Safari 中打开百度首页，并搜索指定关键词“Robot Framework”。通过获取结果网页页面的 Title，判断是否与“robot framework\_百度搜索”相同，如果相同则测试通过，否则失败。

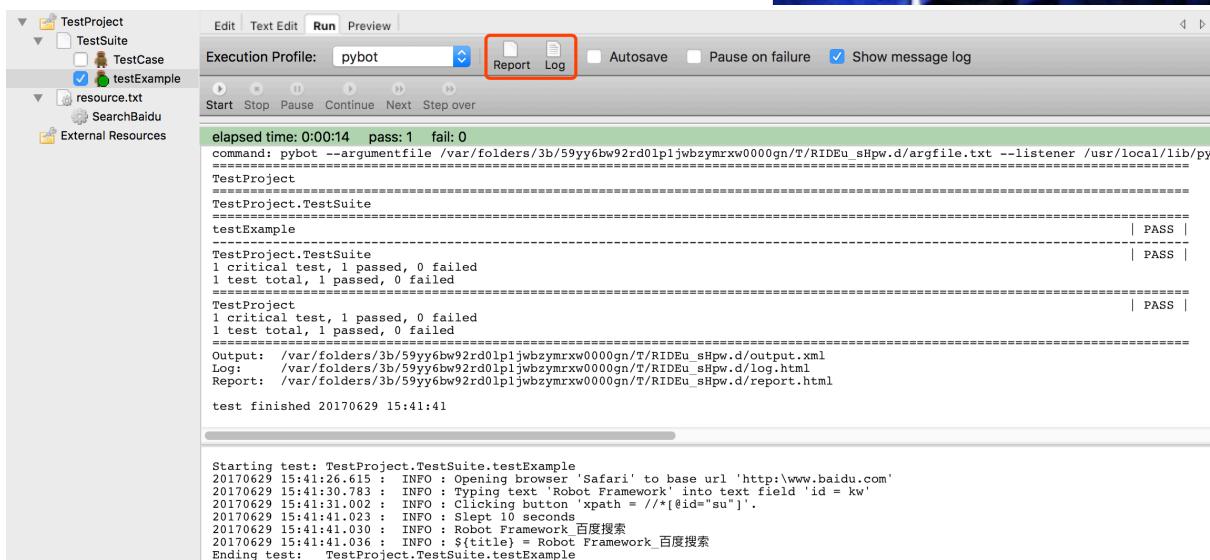
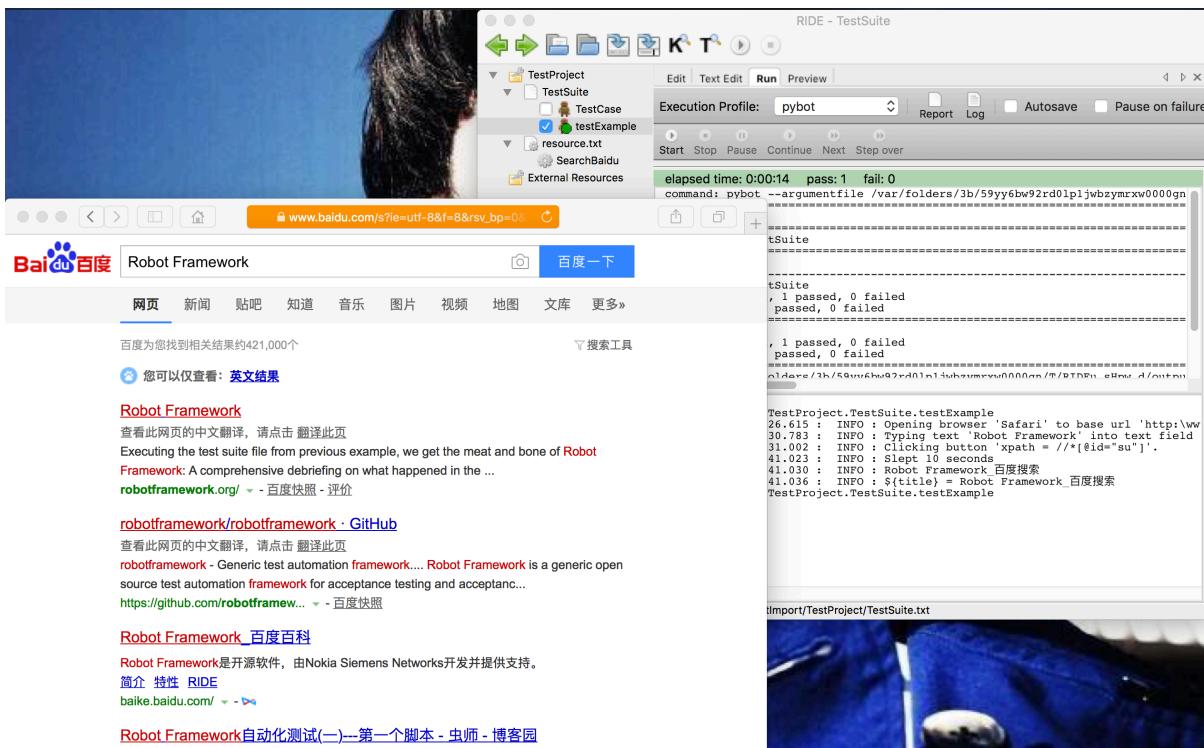
测试用例内容如下图所示：

The screenshot shows the RIDE - TestSuite interface. On the left, the project structure is visible with a checked 'testExample' test case. The main pane displays the test steps:

序号	操作	参数	备注
1	#在Safari中打开百度首页	百度主页网址	确定使用的浏览器
2	Open Browser	http://www.baidu.com/	Safari
3	#等待至百度logo完全显示	通过xpath定位百度logo位置	
4	Wait Until Element is Visible	xpath = //*[@id="lg"]/img	
5	#在搜索框输入“Robot Framework”	通过id定位搜索框	
6	Input Text	id = kw	Robot Framework
7	#点击搜索按钮	通过xpath定位按钮	
8	Click Button	xpath = //*[@id="su"]	
9	#等待10s		
10	Sleep	10s	
11	#记录网页标题		
12	Log Title		
13	#将网页title赋值给\${title}变量		
14	`\${title}`	Get Title	
15	#判断获得的网页标题是否符合期望内容	上一步赋值的变量，内容为获得的网页标题	期望标题中含有的内容
	Should Contain	`\${title}`	robot framework_百度搜索

选中当前用例后，点击 ，或直接按 F8 运行 Test。

测试运行中，会自动打开 Safari 浏览器，并搜索指定关键字。



Test 结束后，无论是否成功通过，都可以点击图中红色框中的按钮，查看 Test Report 及 Test Log。Test Report 更注重脚本的执行结果，而 Test Log 更注重脚本的执行过程。通过点击两种报告右上角的链接，可以直接查看另一个报告。

**TestProject Test Report**

Generated  
20170629 15:54:43 GMT+08:00  
4 minutes 48 seconds ago

**Summary Information**

Status:	All tests passed
Start Time:	20170629 15:54:29.300
End Time:	20170629 15:54:43.347
Elapsed Time:	00:00:14.047
Log File:	<a href="#">log.html</a>

**Test Statistics**

Total Statistics		Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>
All Tests		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>

Statistics by Tag		Total	Pass	Fail	Elapsed	Pass / Fail
No Tags						

Statistics by Suite		Total	Pass	Fail	Elapsed	Pass / Fail
TestProject		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>
TestProject.TestSuite		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>

**Test Details**

Totals Tags Suites Search

Type:  Critical Tests  All Tests

REPORT

## TestProject Test Log

Generated  
20170629 15:54:43 GMT+08:00  
4 minutes 44 seconds ago

### Test Statistics

Total Statistics		Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>
All Tests		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>

Statistics by Tag		Total	Pass	Fail	Elapsed	Pass / Fail
No Tags						

Statistics by Suite		Total	Pass	Fail	Elapsed	Pass / Fail
TestProject		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>
TestProject.TestSuite		1	1	0	00:00:14	<div style="width: 100%; background-color: green;"></div>

### Test Execution Log

- [SUITE] TestProject	00:00:14.047
Full Name:	TestProject
Source:	/Users/QW_Z/Desktop/Intern/Robot/RobotTest/testImport/TestProject
Start / End / Elapsed:	20170629 15:54:29.300 / 20170629 15:54:43.347 / 00:00:14.047
Status:	1 critical test, 1 passed, 0 failed 1 test total, 1 passed, 0 failed
- [SUITE] TestSuite	00:00:14.010
Full Name:	TestProject.TestSuite
Source:	/Users/QW_Z/Desktop/Intern/Robot/RobotTest/testImport/TestProject/TestSuite.txt
Start / End / Elapsed:	20170629 15:54:29.333 / 20170629 15:54:43.343 / 00:00:14.010
Status:	1 critical test, 1 passed, 0 failed 1 test total, 1 passed, 0 failed
- [TEST] testExample	00:00:13.869
Full Name:	TestProject.TestSuite.testExample
Start / End / Elapsed:	20170629 15:54:29.473 / 20170629 15:54:43.342 / 00:00:13.869
Status:	PASS (critical)

## 四. 安装 MongoDB 及 Robot Framework 依赖

### 1. 安装 MongoDB

- 在 MongoDB 下载并安装。

下载地址: <https://www.mongodb.com/download-center#community>

- 创建数据目录

MongoDB 将数据都存在该目录下，但是需要自己创建。

选择想要存放数据目录的文件夹。新建 data 目录，并在 data 目录下再新建一个 db 目录

- 运行 MongoDB 服务器

从 MongoDB 的安装目录下找到 bin 目录，在终端中进入 bin 目录

Window 运行: 将 dbpath 后的路径，改为上一步创建的 data\db 目录所在路径

```
mongod.exe --dbpath E:\data\db
```

OSX 运行:

```
./mongod --dbpath 数据目录路径
```

- 新打开一个终端窗口，在 MongoDB 安装路径下的 bin 目录下，运行 ./mongo

### 2. 安装 pymongo

可以使用 `pip install pymongo`, 或直接本地下载。

下载地址: <https://pypi.python.org/pypi/pymongo/#downloads>

### 3. 下载 MongoDB 依赖库 robotframework-mongodlibrary

下载地址: <https://github.com/iPlantCollaborativeOpenSource/Robotframework-MongoDB-Library>

切换到下载解压包下, 用 `python setup.py install` 命令安装即可

### 4. 在 Robot Framework 的 Test Suite 中导入 MongoDB 依赖库

## 五. Robot Framework 中常用的 MongoDB 关键字

### 1. Connect To MongoDB 连接到数据库

连接时可以指定数据库 Server 地址及端口, 默认为 `localhost:27017`

#连接MongoDB Server	
Connect to Mongodb	

输出结果:

```
15:51:21.981 [INFO] | Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout | dbDocClass | dbTzAware |
| Connect To MondoDB | localhost | 27017 | 10 | None | <type 'dict'> | False |
```

### 2. Get Mongodb Databases 获取连接的数据库 Server 中包含的所有数据库

#获取所连接的server中的所有数据库列表	
@{allDB_list}	Get Mongodb Databases

输出结果:

```
15:51:21.987 [INFO] | @{allDBs} | Get Mongodb Databases |
15:51:21.987 [INFO] | @{allDB_list} = [ admin | local | student ]
```

### 3. Save MongoDB Records 向指定数据库的指定 Collection 中插入新数据

此语句的返回值是插入数据被赋予的 `_id` 值。关键词后依次接: 数据库名, Collection 名, 新插入数据的 JSON 格式。

如果想要存入的数据库或 Collection 不存在, 则新建数据库或 Collection 并进行操作。

#向指定数据库的指定Collection中存入新记录				
\${new_record}	Save Mongodb Records	student	users	{"name": "Jerry", "uid": 1008}

输出结果:

```
15:51:21.992 [INFO] | ${allResults} | Save MongoDB Records | student | users | {'_id': ObjectID('595602f9032a39431fd4b5df'), u'name': u'Jerry', u'uid': 1008} |
15:51:21.992 [INFO] | ${insertResult} = 595602f9032a39431fd4b5df
```

### 4. Update Many MongoDB Records 更新数据

#更新数据					
\${QueryJSON}	Set Variable	{"name": "Tom"}			
\${UpdateJSON}	Set Variable	{"\$set": {"uid": 1002}}			
\${allres}	Update Many Mongodb Records	student	users	\${QueryJSON}	\${UpdateJSON}

其中:

`${QueryJSON}` 定位需要更新的数据 (例子中, 我们要更改 name 为 "Tom" 的记录) ;

`${UpdateJSON}` 定义要更新的内容 (例子中, 我们要将符合条件的记录的 "uid" 改为 1002) ;

Update Many MongoDB Records 后面依次接:

数据库名, Collection 名, \${QueryJSON}, \${UpdateJSON}

输出结果:

```
15:51:21.999 INFO Matched: 1 documents
| ${allResults} | Update Many MongoDB Records | student | users | {u'name': u'Tom'} | {u'$set': {u'uid': 1002}}
15:51:21.999 INFO ${allres} = 1
```

## 5. Retrieve All Mongodb Records 获取所有记录

Retrieve All Mongodb Records + Database 名 + Collection 名

#获取指定数据库Collection中的所有记录				
\$records		Retrieve All Mongodb Records	student	users

如果想更新这条数据并提取这条数据, 可以使用 Retrieve And Update One Mongodb Record.

输出结果:

```
15:51:22.002 INFO ${records} = [{u'_id': ObjectId('595602f2032a39431c4f3d9b')), (u'name', u'Tom'), (u'uid', 1002)]
[(u'_id', ObjectId('595602f9032a39431fd4b5df')), (u'name', u'Jerry'), (u'uid', 1008)]
```

## 6. Retrieve Some Mongodb Records 基于某条件删选数据并提取

Retrieve Some Mongodb Records + 数据库名 + Collection 名 + QueryJSON

Save Mongodb Records	newDB	Students	{"Name": "Jim", "Age": 13, "Uid": 1001}	
Save Mongodb Records	newDB	Students	{"Name": "Alice", "Age": 19, "Uid": 1003}	
Save Mongodb Records	newDB	Students	{"Name": "John", "Age": 26, "Uid": 1004}	
\$records	retrieve all Mongodb records	newDB	Students	
\${some_records}	Retrieve Some Mongodb Records	newDB	Students	{"Age": {"\$gt": 18}}

上图展示的例子中, 我们在名为 newDB 的数据库中新建 Students 表, 并插入三条记录。首先用 Retrieve All Mongodb Records 关键字调取 Students 表中给所有的记录, 结果下图。

```
16:37:08.905 INFO [(u'Age', 13), (u'_id', ObjectId('59575f3462d468452cf118a3')), (u'Name', u'Jim'), (u'Uid', 1001)]
[(u'Age', 19), (u'_id', ObjectId('59575f3462d468452cf118a4')), (u'Name', u'Alice'), (u'Uid', 1003)]
[(u'Age', 26), (u'_id', ObjectId('59575f3462d468452cf118a5')), (u'Name', u'John'), (u'Uid', 1004)]
```

现在我们使用 Retrieve Some Mongodb Records 关键字, 并定义 QueryJSON 为 {"Age": {"\$gt": 18}} ("\$gt": 18 表示大于 18)。得到的结果为:

```
16:37:08.912 INFO [(u'Age', 19), (u'_id', ObjectId('59575f3462d468452cf118a4')), (u'Name', u'Alice'), (u'Uid', 1003)]
[(u'Age', 26), (u'_id', ObjectId('59575f3462d468452cf118a5')), (u'Name', u'John'), (u'Uid', 1004)]
```

如上图所示, 将年龄大于 18 的记录筛选出并调取。

注: JSON 形式表示查询语句

操作	格式	例子
等于	{<key>:<value>}	{"Age": 18} 年龄为 18
小于	{<key>:{ "\$lt": <value> }}	{"Age": {"\$lt": 18}} 年龄小于 18
小于等于	{<key>:{ "\$lte": <value> }}	{"Age": {"\$lte": 18}} 年龄小于或等于 18
大于	{<key>:{ "\$gt": <value> }}	{"Age": {"\$gt": 18}} 年龄大于 18
大于等于	{<key>:{ "\$gte": <value> }}	{"Age": {"\$gte": 18}} 年龄大于或等于 18
不等于	{<key>:{ "\$ne": <value> }}	{"Age": {"\$ne": 18}} 年龄不等于 18

## 7. Get Mongodb Collection Count 获取记录条数

## Get Mongodb Collection Count + Database 名 + Collection 名

#返回指定数据库的指定collection中有几条记录				
\${res}		Get Mongodb Collection Count	student	users

输出结果:

```
15:51:22.004 INFO | ${allResults} | Get MongoDB Collection Count | student | users |
15:51:22.006 INFO ${res} = 2
```

## 8. Remove Mongodb Records 删除数据

Remove Mongodb Records + Database 名 + Collection 名 + 定位需要删除数据的条件语句  
例子中，执行删除 newDB 数据库中 DBUsers 表中“Cid”为 4111 的数据。

\${res}	Remove Mongodb Records	newDB	DBUsers	{"Cid":4111}
---------	------------------------	-------	---------	--------------

输出结果:

```
15:58:27.662 INFO | ${allResults} | Remove MongoDB Records | newDB | DBUsers | {"Cid": 4111} |
15:58:27.663 INFO ${res} = {u'ok': 1.0, u'n': 1}
```

再调出数据库中所有的数据，得到以下结果:

```
16:10:55.549 INFO | ${allResults} | Save MongoDB Records | newDB | DBUsers |
{'_id': ObjectId('5957590f62d46844ecbcba18'), 'Professor': 'Jimmy Choo', 'Cid': 4112, 'Course Name': 'CN'} |
```

## 9. Disconnect From Mongodb 与数据库断开连接

#与数据库断开连接
Disconnect From Mongodb

输出结果:

```
15:43:16.407 INFO | Disconnect From MongoDB |
```

## 10. 删 除 数据 库 或 Collection

删除某数据库: Drop Mongodb Database + 数据库名

删除某 Collection: Drop Mongodb Collection + 数据库名 + Collection 名