第 1 章

初识Robot Framework

Robot Framework 是一款基于 Python 编程语言设计的、可扩展的、关键字驱动模式的测试自动化框架，具备良好的可扩展性，可以通过 XML-RPC 服务扩展支持其他的常用编程语言，可以同时测试多种类型的客户端或者接口，可以支持进行分布式测试执行。

Robot Framework 具体的特点如下：

- 易于使用，采用表格式输入语法以及统一的测试用例（Test Case，也叫测试案例）格式。
- 重用性好，可以利用现有关键字来组合新的用户自定义关键字。
- 支持资源文件，支持多种变量类型，包括字符串变量、List 列表变量、Dictionary 字典变量等。
- 测试用例执行结果报告和日志采用 HTML 格式，易于阅读和邮件转发。
- 提供标签以分类来选择将被执行的测试用例，使得测试用例的选择更加灵活。
- 支持 Web 界面测试、Web 接口服务测试、GUI 测试、多种终端测试。
- 支持多种数据库的操作，包括常用的关系型数据库、非关系型数据库。
- 易于扩展自定义的 Lib 库，可以通过 Python 或者 Java 等其他开发语言来动态扩展 Lib 库。

Robot Framework 自动化测试框架的组成如图 1-0-1 所示。

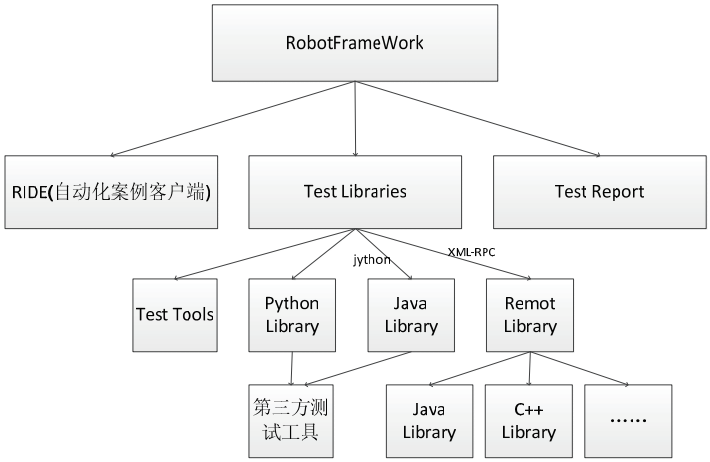

图 1-0-1

- 集成了很多流行的自动化测试工具，比如 Appium、Selenium 等。

- 通过 Jython 的方式，使得以 Python 为主的 Robot Framework 自动化测试框架无缝地与 Java 语言进行完美集成，也可以通过 XML-RPC 远程调用协议来支持 Java 或者 C++ 等流行的编程语言，使对 Python 语言不熟悉的编程爱好者也可以编写自定义的 Library 库。
- 在 Robot Framework 中，使用 Python 语言实现了自动化测试用例编写的客户端 RIDE，使用 RIDE 可以非常简单地完成自动化测试用例的编写，也可以使用 RIDE 完成用户层面的关键字 API 的封装，使得不懂任何编程语言的自动化测试爱好者也可以封装自己的 API 关键字。

Robot Framework 除了提供了我们上面提到的 Ride 外，还提供了很多常用的插件工具，如表 1-0-1 所示。

表 1-0-1　Robot Framework 常用的插件工具

工具插件	说明
Eclipse plugin	Robot Framework 为 Eclipse IDE 开发工具提供的插件，使得用户也可以在 Eclipse 上编写自动化测试用例，GitHub 地址为： https://github.com/NitorCreations/RobotFramework-EclipseIDE/wiki
Robot Plugin for IntelliJ IDEA	和 Eclipse plugin 类似，是为另一个常用的 IDE 开发工具 IntelliJ IDEA 提供的插件，使得用户也可以在 IntelliJ IDEA 上编写自动化测试用例，GitHub 地址为：https://plugins.jetbrains.com/plugin/7430-robot-plugin
Jenkins plugin	这是一个 Jenkins 上使用的插件，这个插件可以使得 Robot Framework 完美地集成在当今非常流行的持续集成工具 Jenkins 上，插件的访问网址为：https://wiki.jenkins-ci.org/display/JENKINS/Robot+Framework+Plugin
Maven plugin	这是 Robot Framework 提供的 maven 仓库插件，可以通过访问网址“http://robotframework.org/MavenPlugin/”获取，当前最新的版本为 1.4.7
Ant task	这是为另一个打包工具 ant 提供的执行插件，使得 Robot Framework 可以通过 ant 的方式来运行。可以通过访问网址“http://code.google.com/p/robotframework-ant”/获取该插件
Pabot	Robot Framework 提供的并发执行器，也就是我们通常说的多线程并发执行模式，可以通过在 Windows 的 cmd 下执行 pip install -U robotframework-pabot 命令进行在线安装，也可以通过访问 GitHub 网址“https://github.com/mkorpela/pabot”进行下载，然后离线进行安装
Atom plugin	Robot Framework 为支持 Atom 而开发的插件，可以通过访问网址“https://atom.io/packages/language-robot-framework”进行下载

1.1 如何创建一个自动化测试项目

一个 Robot Framework 项目其实就和一个我们平时熟知的单元测试项目结构基本是一样的，也包含了测试套件和测试用例的概念。我们可以对 Robot Framework 项目结构做如图 1-1-1 所示的划分。

图 1-1-1

1.1.1　创建测试项目

在 Robot Framework 中，Ride 是一款用 Python 语言实现的用来做自动化测试用例编写的客户端工具。通过访问网址“https://pypi.org/project/robotframework-ride/”即可下载 Ride 工具包进行离线安装，也可以通过在 Windows 的 cmd 命令行中输入“pip install robotframework-ride”进行在线自动安装。安装完成后打开 Ride，选择菜单栏 File→New Project，在 Name 文本框中输入项目名称，此处 Type 我们选择 Directory，单击 OK 按钮，即可创建成功，如图 1-1-2 所示。

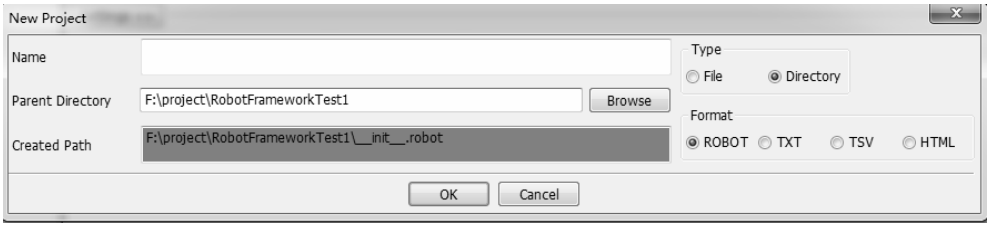

图 1-1-2

存储格式支持多种类型，如表 1-1-1 所示。

表 1-1-1　存储格式支持的类型

储存类型	说明
Type	项目存储方式：文件形式或者目录形式，一般建议选择目录形式
Format	文件存储格式：提供了 ROBOT（默认格式）、TXT、TSV 和 HTML 四种格式

1.1.2　创建测试套件

选择上面我们创建好的项目，右击鼠标键，选择 New Suite 选项，输入测试套件名称，即可创建成功，如图 1-1-3 所示。

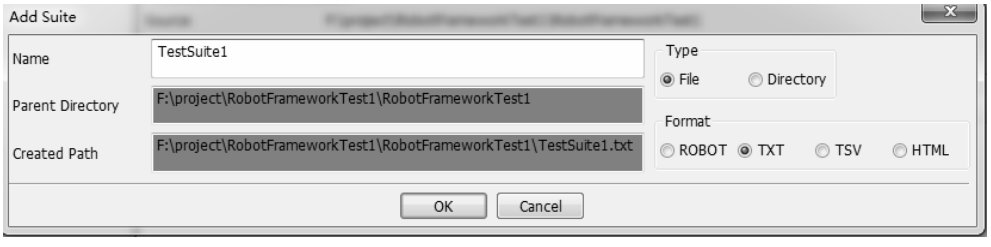

图 1-1-3

1.1.3 创建测试用例

选择上面我们创建好的测试套件，右击，选择 New Test Case 选项，输入用例名称，单击 OK 按钮，即可创建成功，如图 1-1-4 所示。

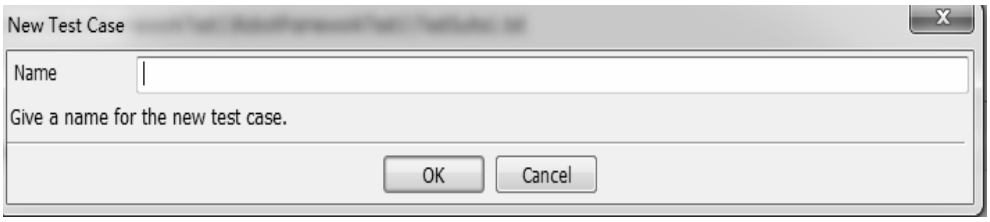

图 1-1-4

创建成功后，即可看到下面的用例编写表格，如图 1-1-5 所示。通过此表格，我们就可以编写测试用例了。

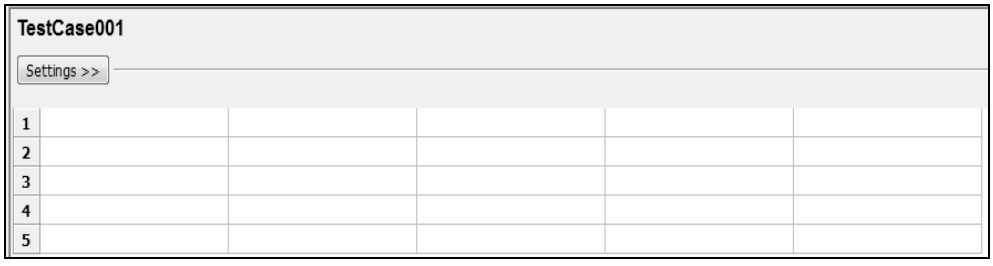

图 1-1-5

1.2 Robot Framework 基础关键字

1.2.1 如何搜索 Robot Framework 的关键字

有两种方式可以快速地打开 RIDE 的关键字搜索对话框。

（1）选择菜单栏中的 Tools→Search Keywords 选项，然后会出现如图 1-2-1 所示的关键字搜索对话框，这个对话框就类似于提供了一个关键字的 API 功能（提供了关键字的名称、关键字的来源库、关键字的使用描述和关键字的参数）。

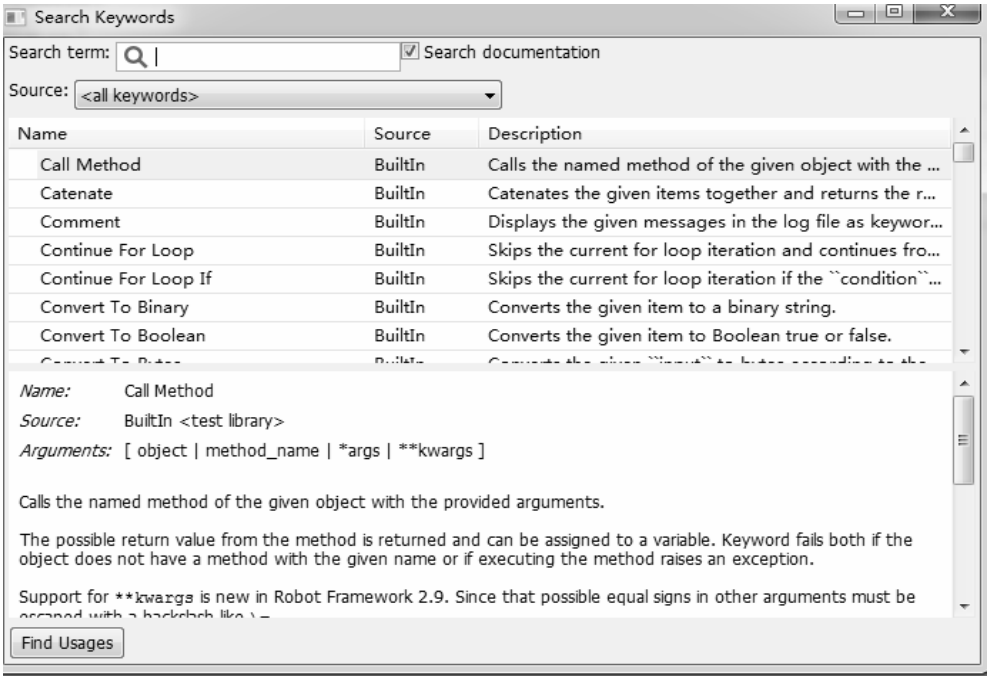

图 1-2-1

（2）直接按 F5 快捷键，就可以自动弹出我们需要的关键字搜索框。

1.2.2　关键字 log

Log 关键字其实就等同于 Python 语言中的 print 函数，可以输出我们想要输出的内容（也就是我们在编程语言中常说的日志输出），比如我们在 test case 中输入如图 1-2-2 所示的内容。

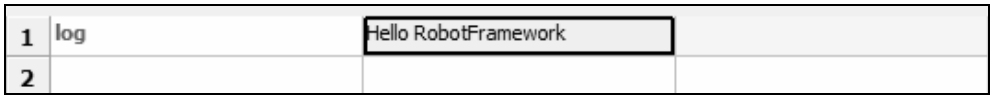

图 1-2-2

勾选我们的测试用例，单击菜单栏 Tools→Run Tests（或者直接快捷键 F8）来执行这条测试用例，如图 1-2-3 所示。

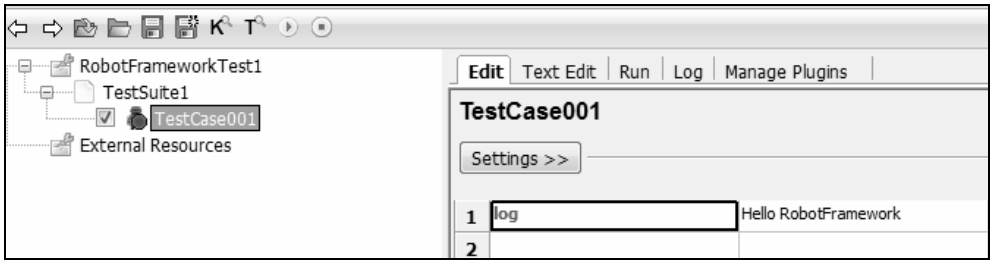

图 1-2-3

执行完成后，切换到 Run 标签，可以看到用例执行的结果。通过运行结果可以看到输出

了我们想要输出的信息 INFO : Hello RobotFramework，如图 1-2-4 所示。

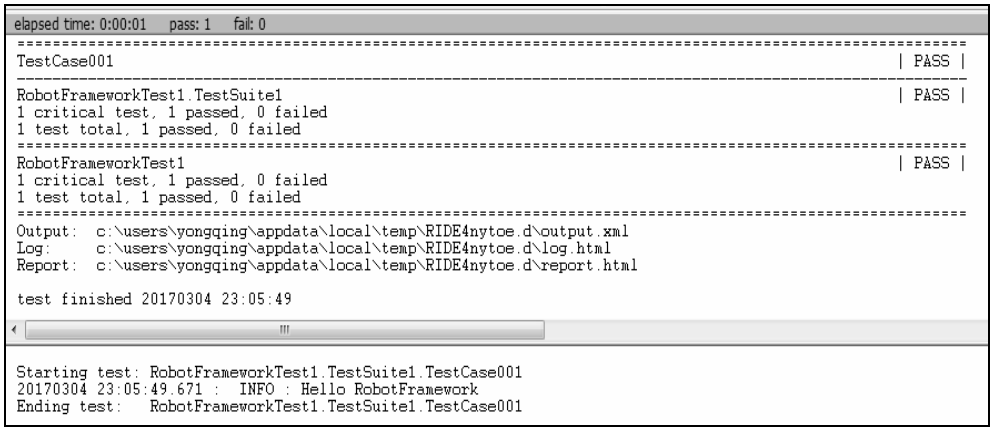

图 1-2-4

1.2.3 如何在用例中定义一个变量

我们可以通过 Set Variable 来定义一个变量，比如我们定义一个变量 var1，并且将这个变量赋值为 Robot，然后将这个变量用 log 输出，如图 1-2-5 所示。

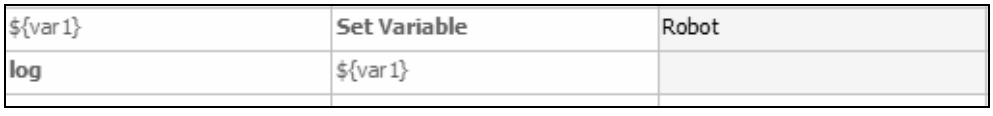

图 1-2-5

执行结果如图 1-2-6 所示。

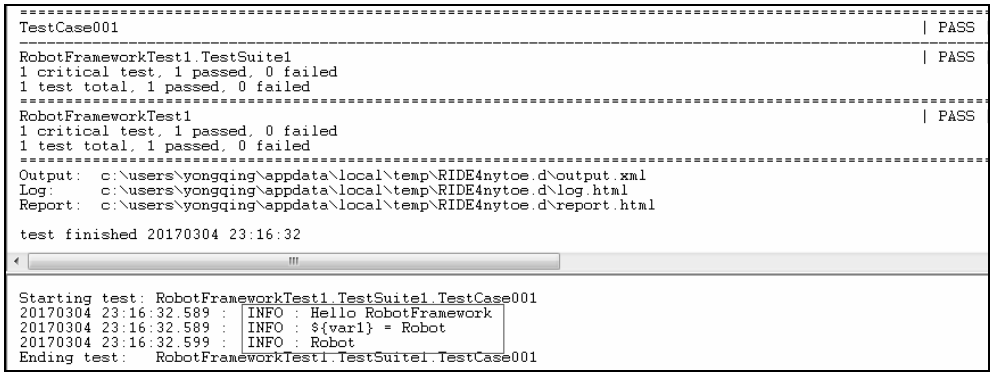

图 1-2-6

1.2.4 如何快速查询某一个关键字的 API 说明

选中关键字，同时按住 Ctrl+Alt 组合键，即可显示该关键字的帮助 API 以及使用示例，如图 1-2-7 所示。

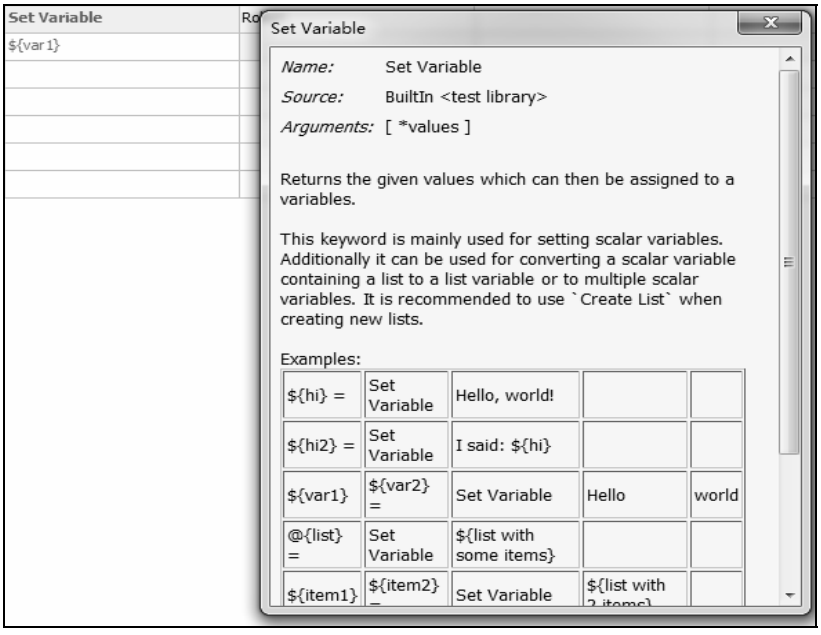

图 1-2-7

1.2.5　如何快速补全关键字

通过键盘输入关键字的前缀，然后同时按住 Ctrl+Alt+空格键，即可快速补全某个关键字，如图 1-2-8 所示。

图 1-2-8

1.2.6　如何定义一个列表

此处我们说的列表，其实就等同于 Python 语言中的列表，是 Python 语言中常用的一种数据结构，也类似于 Java 语言中的 List。

在 Robot Framework 中，我们可以使用 Create List 来创建一个列表，比如我们定义一个列表 list1，并且在创建列表时就添加 3 个元素。然后使用 log 关键字将这个列表中的元素全部输出，如图 1-2-9 所示。

```
@{list1}    Create List hello    robot    framework
log ${list1}
```

1	@{list1}	Create List	hello	robot	framework
2	log	${list1}			

图 1-2-9

执行结果如图 1-2-10 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase2
20170305 22:53:44.087 :  INFO : @{list1} = [ hello | robot | framework ]
20170305 22:53:44.089 :  INFO : [u'hello', u'robot', u'framework']
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase2
```

图 1-2-10

1.2.7 如何定义一个字典

此处我们说的字典其实就等同于 Python 语言中的字典，和列表一样，字典也是 Python 语言中非常常用的一种数据结构，也类似于 Java 语言中的 Map。

在 Robot Framework 中，使用 Create Dictionary 来创建一个字典，比如我们定义一个字典 Dict1，并且在创建字典时就添加两个键值对，然后使用 Log Many 关键字将这个字典中的内容全部输出，如图 1-2-11 所示。

Log Many 关键字类似于 log 关键字，不同的是 log 关键字只可以接收一个参数，而 Log Many 关键字可以同时接收多个参数。

```
&{Dict1}    Create Dictionary  a=hello b=robotframework
Log Many    &{Dict1}
```

1	&{Dict1}	Create Dictionary	a=hello	b=robotframework
2	Log Many	&{Dict1}		

图 1-2-11

执行结果如图 1-2-12 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase003
20170305 23:23:49.439 :  INFO : &{Dict1} = { a=hello | b=robotframework }
20170305 23:23:49.439 :  INFO : a=hello
20170305 23:23:49.439 :  INFO : b=robotframework
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase003
```

图 1-2-12

1.2.8 如何拼接两个字符串

我们可以通过 Catenate 来拼接字符串，比如将“Hello”和“Robot”这两个字符串拼接起来并且输出，如图 1-2-13 所示。

```
${val2} Catenate      Hello  Robot
log ${val2}
```

1	${val2}	Catenate	Hello	Robot
2	log	${val2}		

图 1-2-13

执行结果如图 1-2-14 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase004
20170305 23:31:25.665 :  INFO : ${val2} = Hello Robot
20170305 23:31:25.681 :  INFO : Hello Robot
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase004
```

图 1-2-14

1.2.9　如何使用 for 循环

不管在哪种编程语言中，for 循环都是必不可少的。在 Robot Framework 中，我们也可以使用 for 循环来做遍历处理。

我们可以用 for 循环对一个列表进行遍历，并且输出该列表中的每一个元素。例如，list2 中有 a、b、c、d 四个元素，循环遍历输出这些元素，如图 1-2-15 所示。

```
@{list2}    Create List a   b   c   d
:FOR    ${value}    in @{list2}
    log ${value}
```

@{list2}	Create List	a	b	c	d
:FOR	${value}	in	@{list2}		
	log	${value}			

图 1-2-15

执行结果如图 1-2-16 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase005
20170307 23:18:13.430 :  INFO : @{list2} = [ a | b | c | d ]
20170307 23:18:13.430 :  INFO : a
20170307 23:18:13.430 :  INFO : b
20170307 23:18:13.430 :  INFO : c
20170307 23:18:13.430 :  INFO : d
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase005
```

图 1-2-16

1.2.10　如何中断 for 循环

我们可以使用 Exit For Loop If 关键字来中断一个 for 循环。例如，list2 有 a、b、c、d 四个元素，循环遍历输出这些元素，当输出到元素 c 时跳出这个循环，如图 1-2-17 所示。

```
@{list2}    Create List    a   b   c   d
:FOR    ${value}   in  @{list2}
    log  ${value}
    Exit For Loop If    '${value}'=='c'
```

1	@{list2}	Create List	a	b	c	d
2	:FOR	${value}	in	@{list2}		
3		log	${value}			
4		Exit For Loop If	'${value}'=='c'			

图 1-2-17

执行结果如图 1-2-18 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase006
20170307 23:26:50.877 :  INFO : @{list2} = [ a | b | c | d ]
20170307 23:26:50.877 :  INFO : a
20170307 23:26:50.893 :  INFO : b
20170307 23:26:50.893 :  INFO : c
20170307 23:26:50.893 :  INFO : Exiting for loop altogether.
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase006
```

图 1-2-18

1.2.11 Run Keyword If 判断的使用

Run Keyword If 是一个常用的用来做逻辑判断的关键字，意思是如果满足了某一个判断条件，就会执行关键字。我们在 list3 中放入 0、1、2 三个元素，然后遍历 list3，判断当取到元素 0 时，输出“男生”，如图 1-2-19 所示。

```
@{list3}    Create List 0   1   2
:FOR    ${value}    in  @{list3}
    Run Keyword If  '${value}'= ='0'    log 男生
...
```

1	@{list3}	Create List	0	1	2
2	:FOR	${value}	in	@{list3}	
3		Run Keyword If	'${value}'=='0'	log	男生
4	...				

图 1-2-19

执行结果如图 1-2-20 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase007
20170308 00:00:18.569 :  INFO : @{list3} = [ 0 | 1 | 2 ]
20170308 00:00:18.569 :  INFO : 男生
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase007
```

图 1-2-20

1.2.12 Comment 关键字的使用

Comment 关键字是用来做注释使用的，和很多编程语言中的注释作用一样，可以用来临时注释掉某一行自动化脚本，让其暂时不运行，也可以用来做解释说明使用，如图 1-2-21 所示。

在 Robot Framework 的 RIDE 中，可以选中某一行脚本，右击鼠标键，选择 Comment Rows

选项，然后可以对选中的那一行脚本做注释，注释完成后，这一行脚本将不会再被运行。

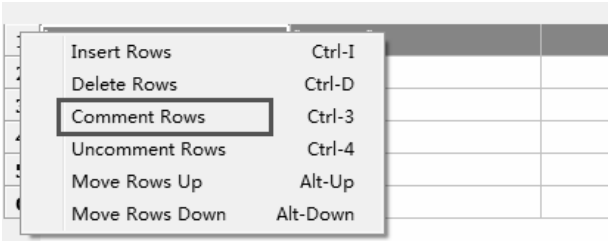

图 1-2-21

```
Comment     log "Comment"
```

如果需要取消注释，右击鼠标键，选择 Uncomment Rows 选项即可。注释取消后，在用例运行时，没有被注释的脚本就会被运行，如图 1-2-22 所示。

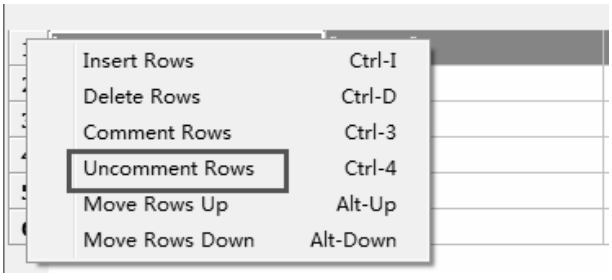

图 1-2-22

1.2.13　Return From Keyword 和 Return From Keyword If 关键字的使用

Return From Keyword 关键字和很多编程语言中的 return 关键字一样，具有如下鲜明的特点：

（1）脚本执行到该关键字后，会直接返回，不会再执行后面的脚本。

（2）返回时会带有对应返回值。调用者可以通过不同的返回值来建立不同的判断分支。

（3）Return From Keyword 关键字一般用于用户自定义关键字中。用户自定义关键字相当于是用系统已有的关键字来封装出一个新的关键字。

（4）Return From Keyword If 关键字用 if 条件来进行判断，当满足指定的 if 条件后，就执行 return 返回。返回时和 Return From Keyword 关键字一样，可以指定返回的具体值。

【示例】我们编写了一个自定义关键字，其中定义了一个入参${valueReturn}，如图 1-2-23 所示，然后通过执行 Return From Keyword If '${value}'=='${valueReturn}' ${value} 来判断我们需要返回的值，如图 1-2-24 所示。

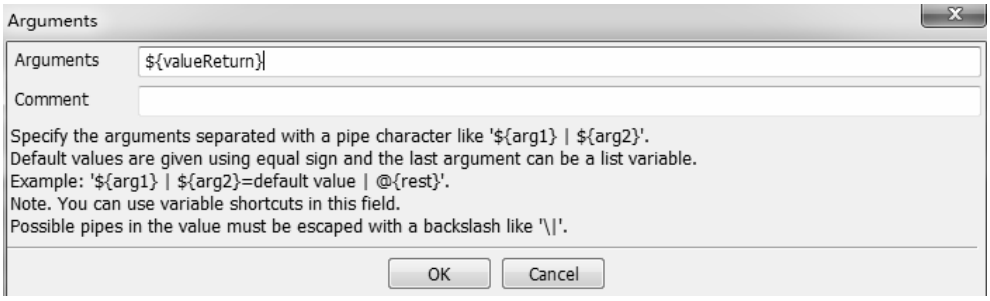

图 1-2-23

```
@{list2}    Create List 1    2    3    4
:FOR    ${value}    IN  @{list2}
    Return From Keyword If    '${value}'=='${valueReturn}'    ${value}
Return From Keyword    ${value}
```

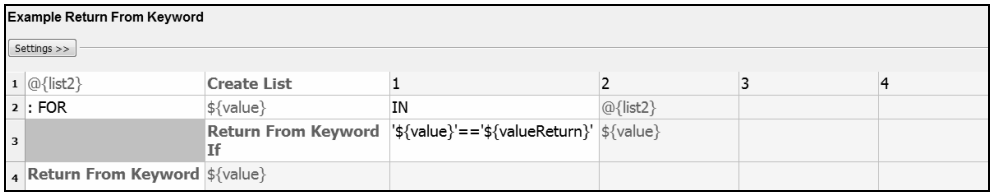

图 1-2-24

自定义关键字完成后，就可以调用了，如图 1-2-25 所示。

${result}	**Example Return From Keyword**	4
log	${result}	

图 1-2-25

执行结果如图 1-2-26 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase030
20180924 11:44:12.210 :  INFO : @{list2} = [ 1 | 2 | 3 | 4 ]
20180924 11:44:12.219 :  INFO : Returning from the enclosing user keyword.
20180924 11:44:12.221 :  INFO : ${result} = 4
20180924 11:44:12.222 :  INFO : 4
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase030
```

图 1-2-26

从执行结果可以看到，在调用 Example Return From Keyword 这个自定义关键字时，我们传入的入参为 4，按照自定义关键字中的判断逻辑返回 4。

1.3 Robot Framework 断言关键字

1.3.1 Should Be Equal 关键字的使用

Should Be Equal 关键字一般用来判断实际结果是否和预期结果相等。例如，我们将变量${value}的值设置为 1，使用 Should Be Equal 关键字来判断${value}是否等于 2，若断言失败，则输出实际值为${value}，和预期不符合，如图 1-3-1 所示。

```
${value}    Set Variable    1
Should Be Equal${value}    2    实际值为${value}，和预期不符合
```

${value}	Set Variable	1	
Should Be Equal	${value}	2	实际值为${value}，和预期不符合

图 1-3-1

执行结果如图 1-3-2 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase011
20170402 13:23:43.228 :  INFO : ${value} = 1
20170402 13:23:43.228 :  FAIL : 实际值为1，和预期不符合: 1 != 2
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase011
```

图 1-3-2

1.3.2 Should Be True 关键字的使用

Should Be True 关键字用来判断返回值是否为 True，例如我们将变量${value}的值同样设置为 1，使用 Should Be True 关键字对表达式'${value}'=='2' 进行 True 和 False 的判断，由于我们设置的值为 1，因此很明显会判断失败，如图 1-3-3 所示。

```
${value}    Set Variable    1
Should Be True  '${value}'=='2' 判断失败
```

${value}	Set Variable	1
Should Be True	'${value}'=='2'	判断失败

图 1-3-3

执行结果如图 1-3-4 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase012
20170402 13:30:58.719 :  INFO : ${value} = 1
20170402 13:30:58.719 :  FAIL : 判断失败
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase012
```

图 1-3-4

1.3.3 Should Contain 关键字的使用

Should Contain 关键字用来判断某个字符串中是否包含了我们预期需要的字符或者字符串，例如我们将变量${str}的值设置为 Robot Framework，使用 Should Contain 关键字来判断${str}是否包含“Hello”这个字符串。很明显，我们执行的结果肯定会判断失败，如图 1-3-5 所示。

```
${str}  Set Variable    RobotFramework
Should Contain ${str}  Hello   字符串${str}中不包含 Hello
```

${str}	Set Variable	RobotFramework	
Should Contain	${str}	Hello	字符串${str}中不包含Hello

图 1-3-5

执行结果如图 1-3-6 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase013
20170402 13:44:52.194 :  INFO : ${str} = RobotFramework
20170402 13:44:52.194 :  FAIL : 字符串RobotFramework中不包含Hello: 'RobotFramework' does not contain 'Hello'
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase013
```

图 1-3-6

1.3.4 Should End With 关键字的使用

Should End With 关键字用来判断某个字符串是否以我们预期指定的字符串来结束，例如我们同样将变量${str}的值设置为 RobotFramework，使用 Should End With 来判断${str}是不是以“Hello”这个字符串来结束。很明显，我们执行的结果肯定会失败，如图 1-3-7 所示。

```
${str}  Set Variable    RobotFramework
Should End With${str}  Hello   字符串${str}中不以 Hello 来结束
```

${str}	Set Variable	RobotFramework	
Should End With	${str}	Hello	字符串${str}中不以Hello来结束

图 1-3-7

执行结果如图 1-3-8 所示。

```
Starting test: RobotFrameworkTest1.TestSuite1.TestCase014
20170402 13:39:15.737 :  INFO : ${str} = RobotFramework
20170402 13:39:15.737 :  FAIL : 字符串RobotFramework中不以Hello来结束: 'RobotFramework' does not end with 'Hello'
Ending test:   RobotFrameworkTest1.TestSuite1.TestCase014
```

图 1-3-8

当我们将“Hello”字符串换成“work”后，再执行一下，会发现执行成功，因为 RobotFramework 是以 work 来结尾的，如图 1-3-9 所示。

```
${str}  Set Variable    RobotFramework
Should End With${str}  work    字符串${str}中不以 Hello 来结束
```

${str}	Set Variable	RobotFramework	
Should End With	${str}	work	字符串${str}中不以Hello来结束

图 1-3-9

执行结果如图 1-3-10 所示。

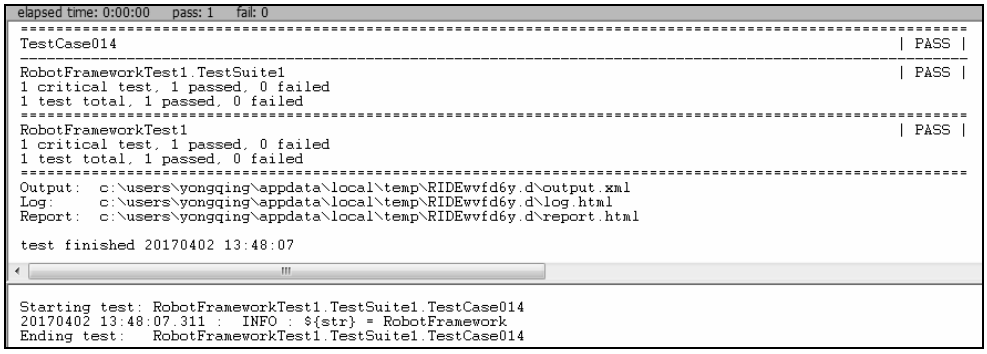

图 1-3-10

1.3.5　其他常用断言关键字

除了我们上面列出的关键字外，Robot Framework 中还提供了大量其他的断言关键字，如表 1-3-1 所示。

表 1-3-1　其他常用断言关键字

<table>
<tr><th>断言关键字</th><th>描述</th></tr>
<tr><td>Should Be Empty</td><td>判断是否为空，若不为空，则执行失败，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>Hello</td></tr>
<tr><td>Should Be Empty</td><td>${value}</td><td>结果不为空</td></tr>
</table></td></tr>
<tr><td>Should Start With</td><td>判断某个字符串是否以预期执行的字符串开始，若以指定的字符串开头，则执行成功，否则执行失败，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>Hello</td><td></td></tr>
<tr><td>Should Start With</td><td>${value}</td><td>qq</td><td>字符串${value}不以 qq 开头</td></tr>
</table></td></tr>
<tr><td>Should Not Start With</td><td>与 Should Start With 刚好相反，若以指定的字符串开头，则执行失败，否则执行成功，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>Hello</td><td></td></tr>
<tr><td>Should Not Start With</td><td>${value}</td><td>qq</td><td>字符串${value}是以 qq 开头</td></tr>
</table></td></tr>
<tr><td>Should Match</td><td>判断某个字符串是否与预期指定的字符串相匹配，若可以匹配，则执行成功，否则执行失败，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>Hello</td><td></td></tr>
<tr><td>Should Match</td><td>${value}</td><td>qq</td><td>字符串${value}不可以匹配 qq</td></tr>
</table></td></tr>
</table>

（续表）

<table>
<tr><th>断言关键字</th><th>描述</th></tr>
<tr><td>Should Not Match</td><td>与 Should Match 刚好相反，若字符串匹配，则执行失败，否则执行成功，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>Hello</td><td></td></tr>
<tr><td>Should Match</td><td>${value}</td><td>Hello</td><td>字符串${value}可以匹配hello</td></tr>
</table></td></tr>
<tr><td>Should Contain X Times</td><td>与 Should Contain 关键字类似,用来判断指定的字符串包含指定的字符或者字符串多少次，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>hello</td><td></td><td></td></tr>
<tr><td rowspan="2">Should Contain X Times</td><td rowspan="2">${value}</td><td rowspan="2">hello</td><td rowspan="2">3</td><td>${value}中没有 3 次包含字</td></tr>
<tr><td>符串 hello</td></tr>
</table></td></tr>
<tr><td>Should Be Equal As Integers</td><td>以整数的形式来进行比较，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td><td></td></tr>
<tr><td>Should Be Equal As Integers</td><td>${value}</td><td>13</td><td>12 和 13 不相等</td></tr>
</table></td></tr>
<tr><td>Should Be Equal As Strings</td><td>以字符串的形式来进行比较，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>q</td><td></td></tr>
<tr><td>Should Be Equal As Strings</td><td>${value}</td><td>13</td><td>q 和 13 不相等</td></tr>
</table></td></tr>
<tr><td>Should Be Equal As Numbers</td><td>以 number 的形式来进行比较，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>1.0</td><td></td></tr>
<tr><td>Should Be Equal As Numbers</td><td>${value}</td><td>1</td><td>1.0 等于 1</td></tr>
</table></td></tr>
<tr><td>Should Not Be Equal</td><td>与 Should Be Equal 用法相反，当带比较的两个值相等时，执行失败，否则执行成功，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>1.0</td><td></td></tr>
<tr><td>Should Not Be Equal</td><td>${value}</td><td>1.0</td><td>1.0 等于 1. 0</td></tr>
</table></td></tr>
<tr><td>Should Not Be Empty</td><td>与 Should Be Empty 用法相反，若为空，则执行失败，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>Hello</td></tr>
<tr><td>Should Not Be Empty</td><td>${value}</td><td>字符串${value}为空</td></tr>
</table></td></tr>
</table>

1.4 BuiltIn 库剩余关键字

1.4.1 常用转换类型关键字

Robot Framework 中提供了很多类型转换关键字，如表 1-4-1 所示。

表 1-4-1　常用转换类型关键字

<table>
<tr><th>转换类型关键字</th><th>描述</th></tr>
<tr><td>Convert To Binary</td><td>将指定的内容转换为二进制形式，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${newvalue}</td><td>Convert To Binary</td><td>${value}</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td></tr>
</table></td></tr>
<tr><td>Convert To Boolean</td><td>将指定的内容转换为布尔类型，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${newvalue}</td><td>Convert To Boolean</td><td>${value}</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td></tr>
</table></td></tr>
<tr><td>Convert To Bytes</td><td>将指定的内容转换为字节数，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td><td></td></tr>
<tr><td>${newvalue}</td><td>Convert To Bytes</td><td>${value}</td><td>int</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Convert To Hex</td><td>将指定的内容转换为十六进制形式，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${newvalue}</td><td>Convert To Hex</td><td>${value}</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td></tr>
</table></td></tr>
<tr><td>Convert To Integer</td><td>将指定的内容转换为 Integer 形式，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${newvalue}</td><td>Convert To Integer</td><td>${value}</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td></tr>
</table></td></tr>
<tr><td>Convert To Number</td><td>将指定的内容转换为 Number 形式，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${newvalue}</td><td>Convert To Number</td><td>${value}</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td></tr>
</table></td></tr>
<tr><td>Convert To Octal</td><td>将指定的内容转换为八进制形式，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${newvalue}</td><td>Convert To Octal</td><td>${value}</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td></tr>
</table></td></tr>
<tr><td>Convert To String</td><td>将指定的内容转换为字符串形式，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${newvalue}</td><td>Convert To String</td><td>${value}</td></tr>
<tr><td>Log</td><td>${newvalue}</td><td></td></tr>
</table></td></tr>
</table>

1.4.2　常用 Get 类型关键字

表 1-4-2 中列出了 Get 类型关键字的常用用法。

表 1-4-2　Get 类型关键字的常用用法

<table>
<tr><th>Get 类型关键字</th><th>描述</th></tr>
<tr><td>Get Count</td><td>获取某个字符串包含指定字符或者字符串的次数，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>hellohello</td><td></td></tr>
<tr><td>${count}</td><td>Get Count</td><td>${value}</td><td>hello</td></tr>
<tr><td>Log</td><td>${count}</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Get Length</td><td>获取指定字符串的长度，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>hellohello</td></tr>
<tr><td>${length}</td><td>Get Length</td><td>${value}</td></tr>
<tr><td>log</td><td>${length}</td><td></td></tr>
</table></td></tr>
<tr><td>Get Time</td><td>获取时间，示例：
<table>
<tr><td>${time}</td><td>Get Time</td><td>format=timestamp</td></tr>
<tr><td>log</td><td>${time}</td><td></td></tr>
</table></td></tr>
<tr><td>Get Variable Value</td><td>获取指定变量的值，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>${result}</td><td>Get Variable Value</td><td>${value}</td></tr>
<tr><td>log</td><td>${result}</td><td></td></tr>
</table></td></tr>
<tr><td>Get Variables</td><td>获取所有的环境变量，示例：
<table>
<tr><td>${vars}</td><td>Get Variables</td></tr>
<tr><td>log</td><td>${vars}</td></tr>
</table></td></tr>
</table>

1.4.3　常用 Import 类型关键字

表 1-4-3 中列出了 Import 类型关键字的常用用法。

表 1-4-3　Import 类型关键字的常用用法

<table>
<tr><th>Import 类型关键字</th><th>描述</th></tr>
<tr><td>Import Library</td><td>在用例中导入某个 Library 库，示例：
<table>
<tr><td>Import Library</td><td>DatabaseLibrary</td></tr>
</table></td></tr>
<tr><td>Import Resource</td><td>在用例中导入某个 Resource 文件，示例：
<table>
<tr><td>Import Resource</td><td>d:\\RUNNING.txt</td></tr>
</table></td></tr>
<tr><td>Import Variables</td><td>在用例中从文件导入变量，示例：
<table>
<tr><td>Import Variables</td><td>variables.py</td></tr>
</table></td></tr>
</table>

1.4.4　常用 Set 类型关键字

表 1-4-4 中列出了 Set 类型关键字的常用用法。

表 1-4-4　Set 类型关键字的常用用法

<table>
<tr><th>Set 类型关键字</th><th>描述</th></tr>
<tr><td>Set Log Level</td><td>设置日志级别，示例：<table><tr><td>Set Log Level</td><td>DEBUG</td></tr></table></td></tr>
<tr><td>Set Variable If</td><td>根据判断条件的结果来确定给某个变量赋值，示例：<table><tr><td>${value}</td><td>Set Variable If</td><td>'2'>'1'</td><td>0</td></tr><tr><td>log</td><td>${value}</td><td></td><td></td></tr></table></td></tr>
<tr><td>Set Global Variable</td><td>设置全局变量，使得该变量也可以在别的用例中使用，示例：<table><tr><td>Set Global Variable</td><td>${book}</td><td>robotframework</td></tr><tr><td>log</td><td>${book}</td><td></td></tr></table></td></tr>
</table>

1.4.5　常用 Run Keyword 类型关键字

表 1-4-5 中列出了 Run Keyword 类型关键字的常用用法。

表 1-4-5　Run Keyword 类型关键字的常用用法

<table>
<tr><th>Run Keyword 类型关键字</th><th>描述</th></tr>
<tr><td>Run Keyword</td><td>执行某个关键字，示例：<table><tr><td>Run Keyword</td><td>log</td><td>RobotFrameWork</td></tr></table></td></tr>
<tr><td>Run Keywords</td><td>执行多个关键字，示例：<table><tr><td>Run Keywords</td><td>Set Global Variable</td><td>${book}</td><td>robotframework</td><td>AND</td><td>log</td><td>${book}</td></tr></table></td></tr>
<tr><td>Run Keyword And Return</td><td>Run Keyword And Return 和上面介绍的 Return From Keyword 和 Return From Keyword If 这两个关键很类似，必须要包装在用户自定义关键字中使用，主要用于执行一个指定关键字并且返回结果，接收[name | *args]多个参数，示例：首先需要定义一个用户自定义关键字，自定义关键字的名称为 Example_RUN_Keyword_AND_RETURN，关键字里面的内容如下：<table><tr><td>Run Keyword And Return</td><td>log</td><td>robotframework</td></tr></table>之后新建一个案例，调用该定义用户关键字：<table><tr><td>Example_RUN_Keyword_AND_RETURN</td></tr></table></td></tr>
</table>

1.4.6　其他关键字

表 1-4-6 中列出了 BuiltIn 库中剩余其他关键字的用法。

表 1-4-6　BuiltIn 库中剩余其他关键字的用法

<table>
<tr><th>关键字</th><th>描述</th></tr>
<tr><td>Evaluate</td><td>调用 Python 中给定的表达式，并且返回结果，示例：
<table>
<tr><td>${value1}</td><td>Evaluate</td><td>int(3)+int(4)</td></tr>
<tr><td>Log</td><td>${value1}</td><td></td></tr>
</table></td></tr>
<tr><td>Fail</td><td>指定某个测试用例在执行某个步骤时直接判定执行失败，示例：
<table>
<tr><td>Fail</td><td>执行失败</td></tr>
</table></td></tr>
<tr><td>Sleep</td><td>按照指定的时间休眠等待，示例：
<table>
<tr><td>Sleep</td><td>3s</td></tr>
</table></td></tr>
<tr><td>Variable Should Exist</td><td>判断某个变量是否存在，若变量存在，则执行成功，否则执行失败，示例：
<table>
<tr><td>Variable Should Exist</td><td>${value}</td><td>变量不存在</td></tr>
</table></td></tr>
<tr><td>Variable Should not Exist</td><td>和 Variable Should Exist 用法相反，若变量存在，则执行失败，否则执行成功，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>12</td></tr>
<tr><td>Variable Should not Exist</td><td>${value}</td><td>变量存在</td></tr>
</table></td></tr>
<tr><td>Wait Until Keyword Succeeds</td><td>在等待的时间内，若关键字执行失败，则按照每隔指定的时间重新执行，若超出等待的时间还执行失败，则执行失败，示例：
<table>
<tr><td>Wait Until Keyword Succeeds</td><td>2 min</td><td>5 sec</td><td>Variable Should Exist</td><td>${value}</td></tr>
</table></td></tr>
<tr><td>Pass Execution</td><td>使用 PASS 状态跳过当前的测试，示例：
<table>
<tr><td>Pass Execution</td><td>Deprecated test.</td></tr>
</table></td></tr>
<tr><td>Replace Variables</td><td>变量替换，示例：
<table>
<tr><td>${value}</td><td>Set Variable</td><td>hello</td></tr>
<tr><td>${result}</td><td>Replace Variables</td><td>${value}</td></tr>
<tr><td>Should Be Equal</td><td>${result}</td><td>hello</td></tr>
</table></td></tr>
</table>

第 2 章

Robot Framework 对数据库的操作

2.1 DatabaseLibrary 库的使用

在自动化过程中，我们经常需要连接不同的数据库，并且对数据库进行很多不同的操作。Robot Framework 提供了 DatabaseLibrary 库来操作数据库。我们可以按照官网中的说明来安装 DatabaseLibrary 库。在浏览器中访问 http://franz-see.github.io/Robotframework-Database-Library/ 页面，即可看到该库的相关安装说明和 API 介绍，如图 2-1-1 所示。

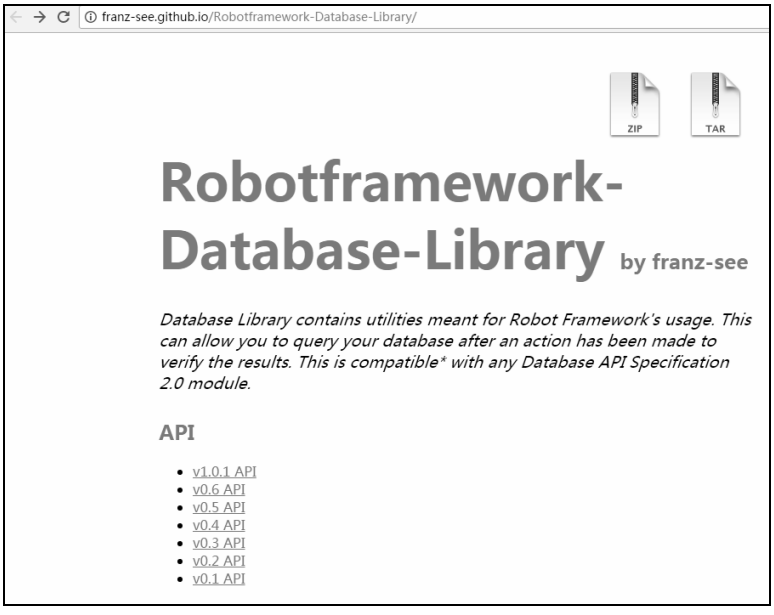

图 2-1-1

可以通过在命令行中执行 pip install -U robotframework-databaselibrary 来进行安装。安装完成后，在使用 DatabaseLibrary 库时，需要预先在测试套件中导入该库，如图 2-1-2 所示。这里以 MySQL 数据库为例，讲述 DatabaseLibrary 库的使用。

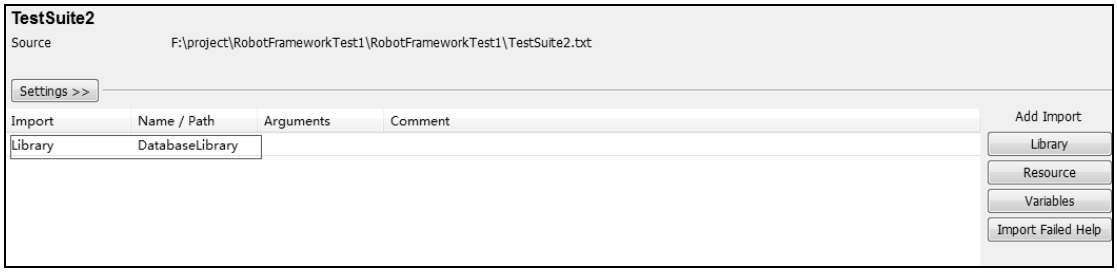

图 2-1-2

要连接到 MySQL，除了要导入 DatabaseLibrary 库外，还需要安装 pure-Python MySQL client library，可以通过访问网址：https://github.com/PyMySQL/PyMySQL 下载该库并且进行安装，或者直接在 cmd 命令行中输入 pip install PyMySQL 来进行安装，如图 2-1-3 所示。

图 2-1-3

2.1.1 如何连接数据库

（1）可以通过 DatabaseLibrary 库中的 Connect To Database 关键字来连接一个 MySQL 数据库。此处以连接本机 MySQL 库为例，如图 2-1-4 所示。

- 数据库用户名：root。
- 数据库密码：root。
- MySQL 数据库端口：3306。
- 数据库名：world。

```
Connect To Database    pymysql    world    root    root    localhost    3306
```

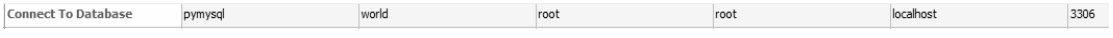

图 2-1-4

执行结果如图 2-1-5 所示。

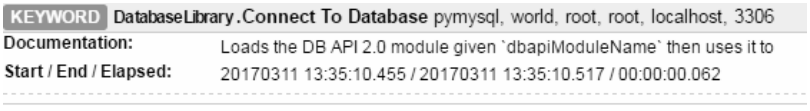

图 2-1-5

（2）还可以通过 Connect To Database Using Custom Params 关键字来连接 MySQL 数据库，如图 2-1-6 所示。

```
Connect To Database Using Custom Params    pymysql
database='world',user='root', password='root', host='localhost', port=3306
```

Connect To Database Using Custom Params	pymysql	database='world', user='root', password='root', host='localhost', port=3306

图 2-1-6

执行结果如图 2-1-7 所示。

KEYWORD DatabaseLibrary.Connect To Database Using Custom Params pymysql, database='world', user='root', password='root', host='localhost', port=3306
Documentation: Loads the DB API 2.0 module given `dbapiModuleName` then uses it to
Start / End / Elapsed: 20170311 13:44:13.825 / 20170311 13:44:13.887 / 00:00:00.062

图 2-1-7

2.1.2　如何断开数据库

可以通过关键字 Disconnect From Database 断开数据库连接，我们在操作数据库时一定不要忘记在操作完成后断开数据库的连接，如图 2-1-8 所示。

```
Connect To Database    pymysql  world  root    root     localhost   3306
Disconnect From Database
```

Connect To Database	pymysql	world	root	root	localhost	3306
Disconnect From Database						

图 2-1-8

执行结果如图 2-1-9 所示。

KEYWORD DatabaseLibrary.Connect To Database pymysql, world, root, root, localhost, 3306
Documentation: Loads the DB API 2.0 module given `dbapiModuleName` then uses it to
Start / End / Elapsed: 20170311 13:38:35.589 / 20170311 13:38:35.651 / 00:00:00.062
KEYWORD DatabaseLibrary.Disconnect From Database
Documentation: Disconnects from the database.
Start / End / Elapsed: 20170311 13:38:35.651 / 20170311 13:38:35.651 / 00:00:00.000

图 2-1-9

2.1.3　如何对数据库的表进行查询

通过 Query 关键字可以对数据库中的表进行查询。此处以查询 MySQL 数据库中某张表的数据为例，我们在 world 数据库中执行“SELECT * FROM city LIMIT 5;”这条 SQL 语句。在 SQL 窗口中查询出来的结果如图 2-1-10 所示。

```
1    SELECT * FROM city LIMIT 5
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

（只读）

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200

图 2-1-10

然后我们使用 Query 关键字来进行查询，如图 2-1-11 所示。

```
Connect To Database Using Custom Params pymysql database='world', user='root',
password='root', host='localhost', port=3306
@{result}   Query    SELECT * FROM city LIMIT 5;
Log Many    @{result}
Disconnect From Database
```

Connect To Database Using Custom Params	pymysql	database='world', user='root', password='root', host='localhost', port=3306
@{result}	Query	SELECT * FROM city LIMIT 5;
Log Many	@{result}	
Disconnect From Database		

图 2-1-11

执行结果如图 2-1-12 所示。

```
KEYWORD DatabaseLibrary.Connect To Database Using Custom Params pymysql, database='world', user='root', password='root', host='localhost', port=3306   00:00:00.078
Documentation:    Loads the DB API 2.0 module given `dbapiModuleName` then uses it to
Start / End / Elapsed:    20170311 13:52:30.159 / 20170311 13:52:30.237 / 00:00:00.078
KEYWORD @{result} = DatabaseLibrary.Query SELECT * FROM city LIMIT 5   00:00:00.000
Documentation:    Uses the input `selectStatement` to query for the values that
Start / End / Elapsed:    20170311 13:52:30.237 / 20170311 13:52:30.237 / 00:00:00.000
13:52:30.237 INFO @{result} = [ (1, 'Kabul', 'AFG', 'Kabol', 1780000) | (2, 'Qandahar', 'AFG', 'Qandahar', 237500) | (3, 'Herat', 'AFG', 'Herat', 186800) | (4, 'Mazar-e-Sharif', 'AFG', 'Balkh', 127800) | (5, 'Amsterdam', 'NLD', 'N...
KEYWORD BuiltIn.Log Many @{result}   00:00:00.000
Documentation:    Logs the given messages as separate entries using the INFO level.
Start / End / Elapsed:    20170311 13:52:30.237 / 20170311 13:52:30.237 / 00:00:00.000
13:52:30.237 INFO (1, 'Kabul', 'AFG', 'Kabol', 1780000)
13:52:30.237 INFO (2, 'Qandahar', 'AFG', 'Qandahar', 237500)
13:52:30.237 INFO (3, 'Herat', 'AFG', 'Herat', 186800)
13:52:30.237 INFO (4, 'Mazar-e-Sharif', 'AFG', 'Balkh', 127800)
13:52:30.237 INFO (5, 'Amsterdam', 'NLD', 'Noord-Holland', 731200)
```

图 2-1-12

2.1.4 如何插入和删除数据

可以通过 Execute Sql String 关键字来执行数据库的插入操作和删除操作。

（1）首先我们来看一下如何向数据库中插入数据。此处以向表 city 中插入一条记录为例，通过 Execute Sql String 关键字来执行 INSERT INTO city(NAME,countrycode,district,population)

VALUES('beijing' ,'ZH','China',217100) ，如图 2-1-13 所示。

```
Connect To Database Using Custom Params    pymysql database='world',
user='root', password='root', host='localhost', port=3306
Execute Sql String INSERT INTO city(NAME,countrycode,district,population)
VALUES('beijing' ,'ZH','China',217100)
Disconnect From Database
```

Connect To Database Using Custom Params	pymysql	database='world', user='root', password='root', host='localhost', port=3306
Execute Sql String	INSERT INTO city(NAME,countrycode,district,po VALUES('beijing' ,'ZH','China',217100)	
Disconnect From Database		

图 2-1-13

执行结果如图 2-1-14 所示。

```
KEYWORD DatabaseLibrary.Connect To Database Using Custom Params pymysql, database='world', user='root', password='root', host='localhost', port=3306
Documentation:        Loads the DB API 2.0 module given `dbapiModuleName` then uses it to
Start / End / Elapsed: 20170311 14:11:04.375 / 20170311 14:11:04.422 / 00:00:00.047
KEYWORD DatabaseLibrary.Execute Sql String INSERT INTO city(NAME,countrycode,district,population) VALUES('beijing' ,'ZH','China',217100)
Documentation:        Executes the sqlString as SQL commands.
Start / End / Elapsed: 20170311 14:11:04.422 / 20170311 14:11:04.422 / 00:00:00.000
KEYWORD DatabaseLibrary.Disconnect From Database
Documentation:        Disconnects from the database.
Start / End / Elapsed: 20170311 14:11:04.422 / 20170311 14:11:04.422 / 00:00:00.000
```

图 2-1-14

在 SQL 窗口查询刚刚执行的 insert 语句是否执行成功。我们可以看到已经成功插入了数据，如图 2-1-15 所示。

```
SELECT * FROM city WHERE NAME='beijing'
```

1 结果　2 个配置文件　3 信息　4 表数据　5 信息

（只读）

ID	Name	CountryCode	District	Population
4081	beijing	ZH	China	217100

图 2-1-15

（2）然后我们看一下怎么删除表中的数据。我们将上面插入的“'beijing' ,'ZH','China', 217100”这条数据从数据库中删除，如图 2-1-16 所示。

```
Connect To Database Using Custom Paramspymysql database='world', user='root',
password='root', host='localhost', port=3306
Execute Sql String delete from city where NAME='beijing'
Disconnect From Database
```

Connect To Database Using Custom Params	pymysql	database='world', user='root', password='root', host='localhost', port=3306
Execute Sql String	delete from city where NAME='beijing'	
Disconnect From Database		

图 2-1-16

执行结果如图 2-1-17 所示。

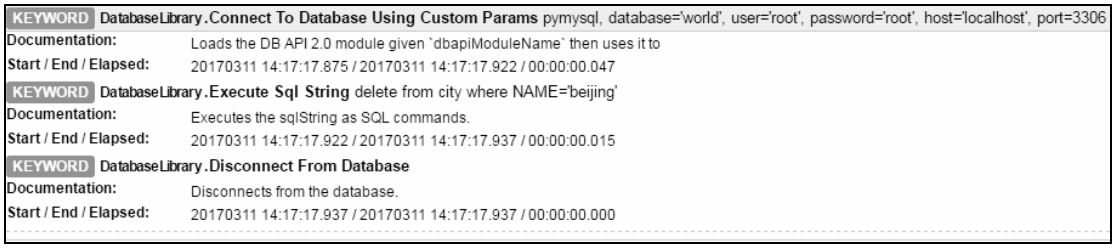

图 2-1-17

在 SQL 窗口查询一下有没有将数据成功删除。从查询的结果看，数据已经成功地被删除，如图 2-1-18 所示。

图 2-1-18

2.1.5 如何执行数据库脚本文件

在做自动化测试时，我们经常需要构造数据或者对库中的数据进行初始化，但是如果我们每次都是将要执行的数据库脚本按条写在用例中，那么将非常不好维护，因此我们需要直接执行数据库脚本文件。在 DatabaseLibrary 库中，可以通过 Execute Sql Script 关键字来执行数据库脚本文件。

此处以执行本地磁盘中的 script.sql 为例。在 script.sql 脚本中放入需要执行的语句，如图 2-1-19 所示。

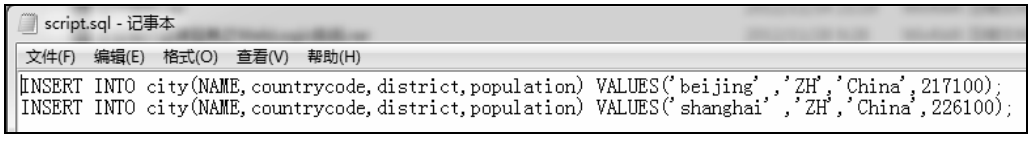

图 2-1-19

完整示例如图 2-1-20 所示。

```
Connect To Database Using Custom Params    pymysql
    database='world',user='root', password='root', host='localhost', port=3306
Execute Sql Script  f:/script.sql
Disconnect From Database
```

Connect To Database Using Custom Params	pymysql	database='world', user='root', password='root', host='localhost', port=3306
Execute Sql Script	f:/script.sql	
Disconnect From Database		

图 2-1-20

执行结果如图 2-1-21 所示。

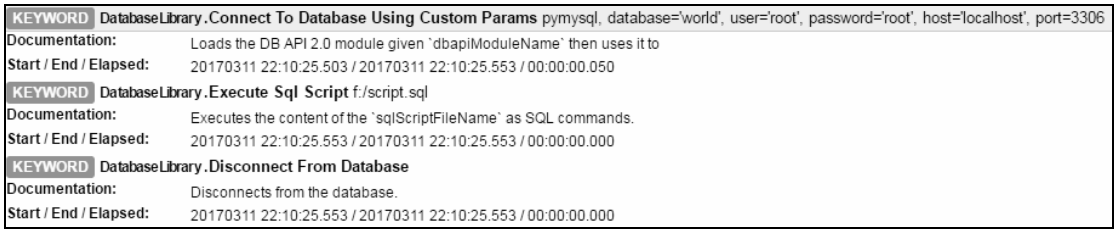

图 2-1-21

执行成功后，对数据库进行查询，会发现脚本已经执行成功、数据已经成功插入，如图 2-1-22 所示。

图 2-1-22

2.1.6　DatabaseLibrary 库的其他操作关键字

表 2-1-1 中描述了 DatabaseLibrary 库中其他关键字的使用方法。

表 2-1-1　DatabaseLibrary 库其他关键字的使用方法

关键字	使用描述
Check If Exists In Database	检查数据库查询是否有返回结果，若有返回结果，则用例执行成功，否则执行失败，示例： Check If Exists In Database　SELECT * FROM city WHERE NAME='beijing' OR NAME='shanghai'

（续表）

<table>
<tr><th>关键字</th><th>使用描述</th></tr>
<tr><td>Check If Not Exists In Database</td><td>检查数据库查询是否有返回结果，若有返回结果，则用例执行失败，否则执行成功，示例：<table><tr><td>Check If Not Exists In Database</td><td>SELECT * FROM city WHERE NAME='beijing' and NAME='shanghai'</td></tr></table></td></tr>
<tr><td>Delete All Rows From Table</td><td>删除数据库中某张表中的全部数据，示例：<table><tr><td>Delete All Rows From Table</td><td>World</td></tr></table></td></tr>
<tr><td>Description</td><td>描述数据库的查询结果，示例：<table><tr><td>@{result}</td><td>Description</td><td>SELECT * FROM city WHERE NAME='beijing' or NAME='shanghai'</td></tr><tr><td>Log Many</td><td>@{result}</td><td></td></tr></table></td></tr>
<tr><td>Row Count</td><td>统计 SQL 查询返回的记录数，示例：<table><tr><td>${rowCount}</td><td>Row Count</td><td>SELECT * FROM city WHERE NAME='beijing' or NAME='shanghai'</td></tr><tr><td>Log</td><td>${rowCount}</td><td></td></tr></table></td></tr>
<tr><td>Row Count Is 0</td><td>检查 SQL 查询返回的记录数是否为 0，示例：<table><tr><td>Row Count Is 0</td><td>SELECT * FROM city WHERE NAME='beijing' or NAME='shanghai'</td></tr></table></td></tr>
<tr><td>Row Count Is Equal To X</td><td>检查 SQL 查询返回的记录数是否等于某个值，示例：<table><tr><td>Row Count Is Equal To X</td><td>SELECT * FROM city WHERE NAME='beijing' or NAME='shanghai'</td><td>1</td></tr></table></td></tr>
<tr><td>Row Count Is Greater Than X</td><td>检查 SQL 查询返回的记录数是否大于某个值，示例：<table><tr><td>Row Count Is Greater Than X</td><td>SELECT * FROM city WHERE NAME='beijing' or NAME='shanghai'</td><td>1</td></tr></table></td></tr>
<tr><td>Row Count Is Less Than X</td><td>检查 SQL 查询返回的记录数是否小于某个值，示例：<table><tr><td>Row Count Is Less Than X</td><td>SELECT * FROM city WHERE NAME='beijing' or NAME='shanghai'</td><td>1</td></tr></table></td></tr>
<tr><td>Table Must Exist</td><td>判断数据库中表是否存在，示例：<table><tr><td>Table Must Exist</td><td>city</td></tr></table></td></tr>
</table>

2.2 MongoDBLibrary 库的使用

MongoDB 是非常常用的一个非关系型数据库。Robot Framework 提供了对 MongoDB 数据库测试操作的支持。我们可以通过在浏览器中访问 GitHub 的网站地址 https://github.com/iPlant

CollaborativeOpenSource/Robotframework-MongoDB-Library，查看该库的相关安装说明和 API 介绍，如图 2-2-1 所示。

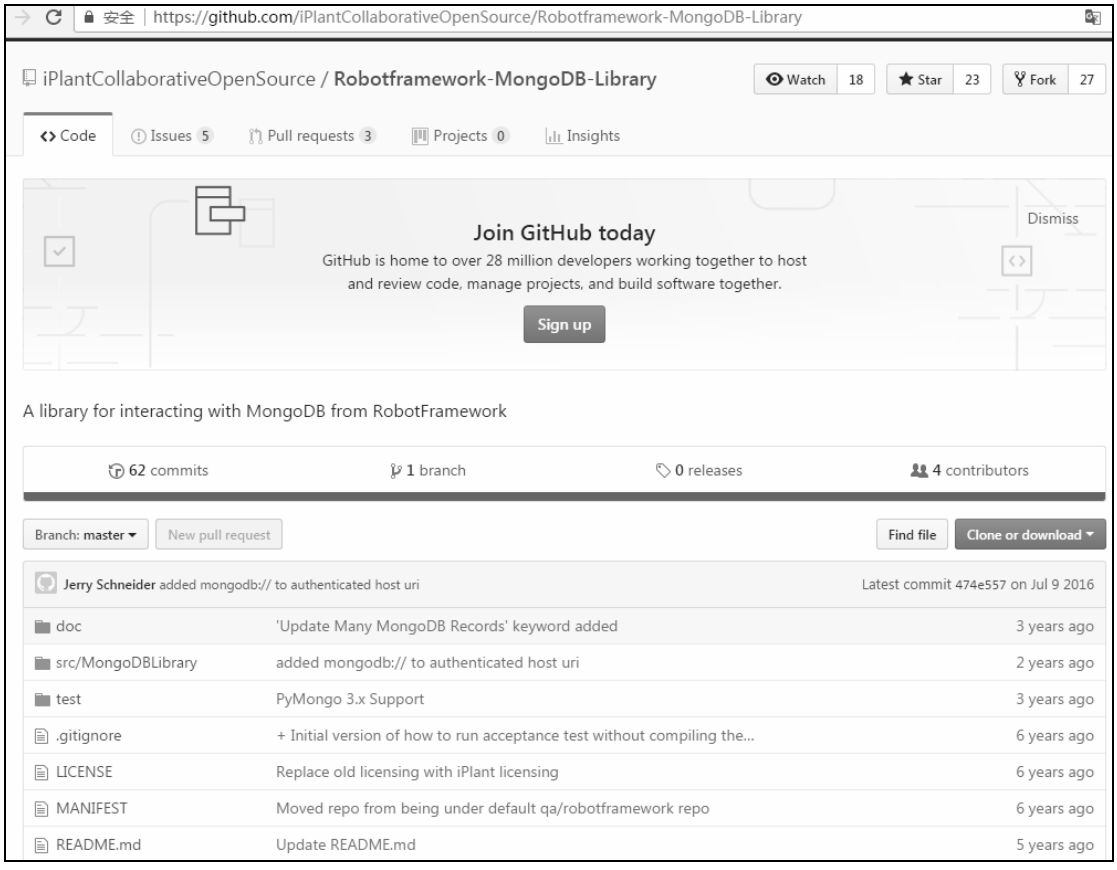

图 2-2-1

安装完成后，在使用 MongoDBLibrary 库时，需要预先在测试套件中导入该库，如图 2-2-2 所示。

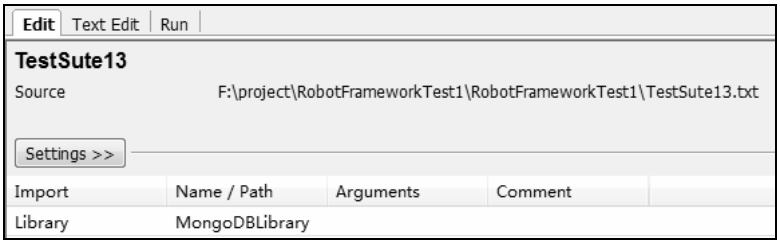

图 2-2-2

2.2.1　MongoDB 数据库的连接和断开

在 MongoDBLibrary 中通过 Connect To MongoDB 关键字来连接到 MongoDB 数据库，该

关键字接收[dbHost=localhost | dbPort=27017 | dbMaxPoolSize=10 | dbNetworkTimeout=None | dbDocClass= | dbTZAware=False]六个参数。其中，dbHost 参数指的是 MongoDB 数据库的 IP 地址，dbPort 参数指的是 MongoDB 数据库的端口号，不输入时默认为 27017；dbMaxPoolSize 参数指的是数据库连接的最大线程池大小，不输入时默认大小为 10。

【示例 1】我们连接到本地电脑上一个已经启动好的 MongoDB 数据库上，这里预先启动了一个 3.2 版本的 MongoDB 数据库，如图 2-2-3 所示。

图 2-2-3

在 RIDE 中，使用 Connect To MongoDB 来连接刚刚启动好的数据库，如图 2-2-4 所示。

图 2-2-4

运行结果如图 2-2-5 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase001
20180825 17:47:32.470 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase001
```

图 2-2-5

执行完成后，查看一下 MongoDB 服务端的日志。从如图 2-2-6 所示的 MongoDB 服务端的日志可以看到，已经成功和 MongoDB 数据库建立了连接。

```
2018-08-25T17:45:30.134+0800 I CONTROL  [main] Hotfix KB2731284 or later update
is not installed, will zero-out data files
2018-08-25T17:45:30.138+0800 I CONTROL  [initandlisten] MongoDB starting :
pid=8964 port=27017 dbpath=D:\MongoDB\Server\3.2\data 64-bit host=yongqing-PC
2018-08-25T17:45:30.138+0800 I CONTROL  [initandlisten] targetMinOS: Windows
Vista/Windows Server 2008
2018-08-25T17:45:30.139+0800 I CONTROL  [initandlisten] db version v3.2.4
2018-08-25T17:45:30.139+0800 I CONTROL  [initandlisten] git version:
e2ee9ffcf9f5a94fad76802e28cc978718bb7a30
2018-08-25T17:45:30.140+0800 I CONTROL  [initandlisten] allocator: tcmalloc
```

```
2018-08-25T17:45:30.140+0800 I CONTROL  [initandlisten] modules: none
2018-08-25T17:45:30.140+0800 I CONTROL  [initandlisten] build environment:
2018-08-25T17:45:30.161+0800 I CONTROL  [initandlisten]     distarch: x86_64
2018-08-25T17:45:30.162+0800 I CONTROL  [initandlisten]     target_arch: x86_64
2018-08-25T17:45:30.162+0800 I CONTROL  [initandlisten] options: { storage:
{ dbPath: "D:\MongoDB\Server\3.2\data" }, systemLog: { destination: "file", path:
"D:\MongoDB\Server\3.2\logs\log.log" } }
2018-08-25T17:45:30.164+0800 I STORAGE  [initandlisten] wiredtiger_open config:
create,cache_size=4G,session_max=20000,eviction=(threads_max=4),config_base=f
alse,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor
=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2
GB),statistics_log=(wait=0),
2018-08-25T17:45:31.202+0800 I FTDC     [initandlisten] Initializing full-time
diagnostic data capture with directory
'D:/MongoDB/Server/3.2/data/diagnostic.data'
2018-08-25T17:45:31.202+0800 I NETWORK  [HostnameCanonicalizationWorker]
Starting hostname canonicalization worker
2018-08-25T17:45:31.359+0800 I NETWORK  [initandlisten] waiting for connections
on port 27017
2018-08-25T17:46:53.205+0800 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:2621 #1 (1 connection now open)
2018-08-25T17:46:53.730+0800 I NETWORK  [conn1] end connection 127.0.0.1:2621 (0
connections now open)
2018-08-25T17:47:32.471+0800 I NETWORK  [initandlisten] connection accepted from
127.0.0.1:2650 #2 (1 connection now open)
2018-08-25T17:47:33.031+0800 I NETWORK  [conn2] end connection 127.0.0.1:2650 (0
connections now open)
```

图 2-2-6

在 MongoDBLibrary 中通过 Disconnect From Mongodb 关键字来断开已经建立的 MongoDB 数据库连接。

【示例 2】通过 Disconnect From Mongodb 关键字断开 MongoDB 的数据库连接，如图 2-2-7 所示。

Connect To MongoDB	127.0.0.1	27017	2
Disconnect From Mongodb			

图 2-2-7

运行结果如图 2-2-8 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase001
20180825 17:57:04.680 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
20180825 17:57:04.684 :  INFO : | Disconnect From MongoDB |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase001
```

图 2-2-8

从 MongoDB 的日志可以看到，在执行 Disconnect From Mongodb 关键字操作后，数据库服务端日志中已经显示数据库连接终止。

```
[conn3] end connection 127.0.0.1:2945 (0 connections now open)
```

2.2.2 Get Mongodb Databases 和 Get Mongodb Collections

在 MongoDBLibrary 中通过 Get Mongodb Databases 关键字来获取当前 MongoDB 下所有在用的数据库。

【示例 1】我们通过 Get Mongodb Databases 关键字来获取上面启动的 MongoDB 下的所有数据库，如图 2-2-9 所示。

Connect To MongoDB	127.0.0.1	27017	2
@{DBs}	**Get Mongodb Databases**		
Log Many	@{DBs}		
Disconnect From Mongodb			

图 2-2-9

运行结果如图 2-2-10 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase003
20180825 21:55:34.867 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
20180825 21:55:34.873 :  INFO : | @{allDBs} | Get Mongodb Databases |
20180825 21:55:34.873 :  INFO : @{DBs} = [ local ]
20180825 21:55:34.875 :  INFO : local
20180825 21:55:34.877 :  INFO : | Disconnect From MongoDB |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase003
```

图 2-2-10

从运行结果可以看到只获取到了一个名叫 local 的数据库。我们通过客户端连接到 MongoDB 服务端，然后执行 show databases 命令，可以看到得到的结果和我们通过 Get Mongodb Databases 关键字来获取到的数据库信息是一致的，如图 2-2-11 所示。

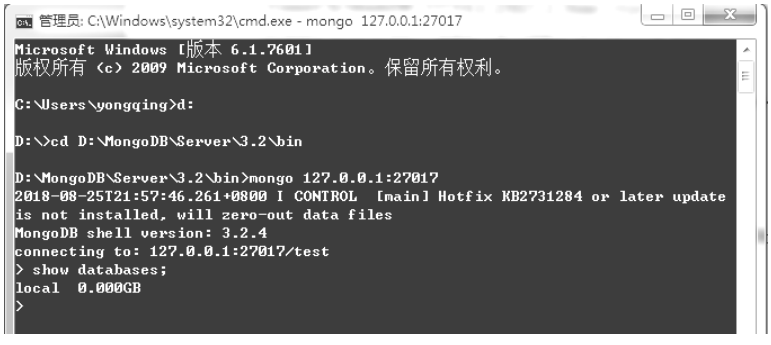

图 2-2-11

通过 Get Mongodb Collections 关键字可以获取到指定 MongoDB 数据库下的所有 Collection，该关键字接收[dbName]一个参数。

【示例 2】我们通过 Get Mongodb Collections 关键字来获取到 local 库下的所有 Collection，如图 2-2-12 所示。

Connect To MongoDB	127.0.0.1	27017	2
@{DBs}	**Get Mongodb Databases**		
Log Many	@{DBs}		
@{allCollections}	**Get Mongodb Collections**	local	
Log Many	@{allCollections}		
Disconnect From Mongodb			

图 2-2-12

运行结果如图 2-2-13 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase002
20180825 22:03:31.189 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
20180825 22:03:31.197 :  INFO : | @{allDBs} | Get Mongodb Databases |
20180825 22:03:31.198 :  INFO : @{DBs} = [ local ]
20180825 22:03:31.200 :  INFO : local
20180825 22:03:31.205 :  INFO : | @{allCollections} | Get MongoDB Collections |
local |
20180825 22:03:31.206 :  INFO : @{allCollections} = [ startup_log ]
20180825 22:03:31.208 :  INFO : startup_log
20180825 22:03:31.210 :  INFO : | Disconnect From MongoDB |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase002
```

图 2-2-13

从运行结果可以看到，获取到的 local 库下的 Collection 名叫 startup_log，然后我们通过客户端连接到服务端，通过客户端 show collections 命令来获取 Collection。我们可以看到获取到的 Collection 是完全一致的，如图 2-2-14 所示。

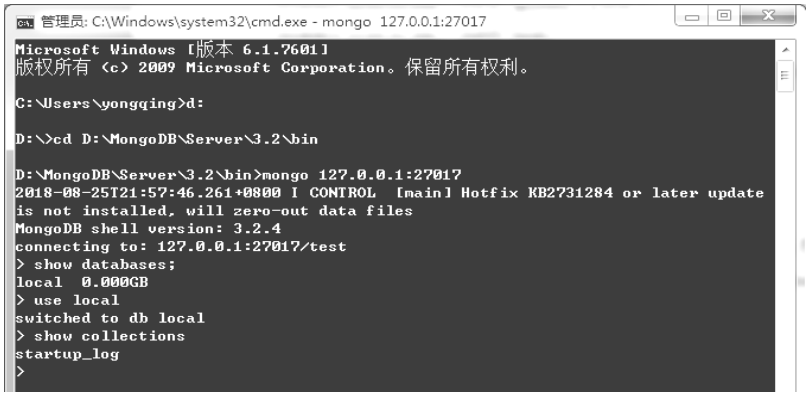

图 2-2-14

2.2.3 Save Mongodb Records

Save MongoDB Records 关键字用来向指定的 Collection 中保存插入的记录，接收[dbName | dbCollName | recordJSON]三个参数。

【示例】我们向 startup_log 这个 Collection 中插入一条记录，如图 2-2-15 所示。

Connect To MongoDB	127.0.0.1	27017	2
Save Mongodb Records	local	startup_log	{"book":"RobotFramework"}
Disconnect From Mongodb			

图 2-2-15

运行结果如图 2-2-16 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase004
20180825 22:16:34.333 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
20180825 22:16:34.344 :  INFO : | ${allResults} | Save MongoDB Records | local
| startup_log | {u'book': u'RobotFramework', '_id':
ObjectId('5b8164c2685b132ec4739503')} |
20180825 22:16:34.347 :  INFO : | Disconnect From MongoDB |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase004
```

图 2-2-16

执行完成后，我们通过 MongoDB 客户端连接到服务端，执行 db.startup_log.find()命令来查看 startup_log 这个 Collection 下的记录。可以看到{"book":"RobotFramework"}这条数据记录已经成功插入 MongoDB 中，如图 2-2-17 所示。

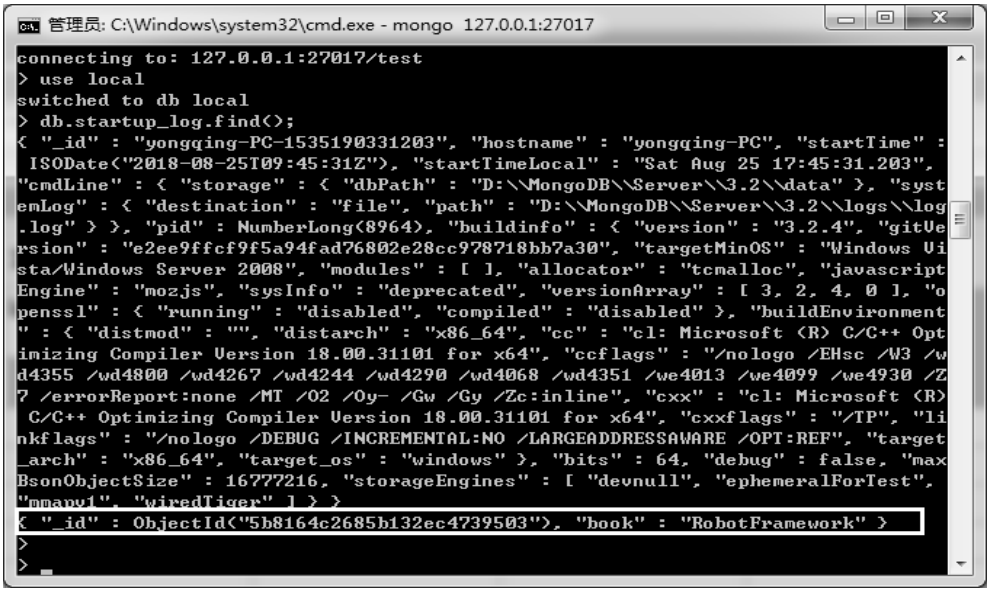

图 2-2-17

2.2.4　Retrieve All Mongodb Records

Retrieve All Mongodb Records 关键字用来获取指定 Collection 下的所有数据记录，接收[dbName | dbCollName | returnDocuments=False]三个参数。

【示例】我们通过 Retrieve All Mongodb Records 关键字来获取 startup_log 下的数据记录，如图 2-2-18 所示。

Connect To MongoDB	127.0.0.1	27017	2	
${allResults}	**Retrieve All Mongodb Records**	local	startup_log	True
log	${allResults}			
Disconnect From Mongodb				

图 2-2-18

运行结果如图 2-2-19 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase005
20180825 22:38:05.197 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
20180825 22:38:05.213 :  INFO : ${allResults} = [{u'hostname': u'yongqing-PC',
u'pid': 8964L, u'startTimeLocal': u'Sat Aug 25 17:45:31.203', u'cmdLine':
{u'storage': {u'dbPath': u'D:\\MongoDB\\Server\\3.2\\data'}, u'systemLog':
{u'path': u'D:\\Mong...
20180825 22:38:05.215 :  INFO : [{u'hostname': u'yongqing-PC', u'pid': 8964L,
u'startTimeLocal': u'Sat Aug 25 17:45:31.203', u'cmdLine': {u'storage':
{u'dbPath': u'D:\\MongoDB\\Server\\3.2\\data'}, u'systemLog': {u'path':
u'D:\\MongoDB\\Server\\3.2\\logs\\log.log', u'destination': u'file'}},
u'startTime': datetime.datetime(2018, 8, 25, 9, 45, 31), u'_id':
u'yongqing-PC-1535190331203', u'buildinfo': {u'storageEngines': [u'devnull',
u'ephemeralForTest', u'mmapv1', u'wiredTiger'], u'maxBsonObjectSize': 16777216,
u'bits': 64, u'sysInfo': u'deprecated', u'modules': [], u'openssl': {u'compiled':
u'disabled', u'running': u'disabled'}, u'javascriptEngine': u'mozjs',
u'version': u'3.2.4', u'gitVersion':
u'e2ee9ffcf9f5a94fad76802e28cc978718bb7a30', u'versionArray': [3, 2, 4, 0],
u'debug': False, u'buildEnvironment': {u'cxxflags': u'/TP', u'cc': u'cl:
Microsoft (R) C/C++ Optimizing Compiler Version 18.00.31101 for x64',
u'linkflags': u'/nologo /DEBUG /INCREMENTAL:NO /LARGEADDRESSAWARE /OPT:REF',
u'distarch': u'x86_64', u'cxx': u'cl: Microsoft (R) C/C++ Optimizing Compiler
Version 18.00.31101 for x64', u'ccflags': u'/nologo /EHsc /W3 /wd4355 /wd4800
/wd4267 /wd4244 /wd4290 /wd4068 /wd4351 /we4013 /we4099 /we4930 /Z7
/errorReport:none /MT /O2 /Oy- /Gw /Gy /Zc:inline', u'target_arch': u'x86_64',
u'distmod': u'', u'target_os': u'windows'}, u'targetMinOS': u'Windows
Vista/Windows Server 2008', u'allocator': u'tcmalloc'}}, {u'_id':
ObjectId('5b8164c2685b132ec4739503'), u'book': u'RobotFramework'}]
20180825 22:38:05.217 :  INFO : | Disconnect From MongoDB |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase005
```

图 2-2-19

如图 2-2-20 所示，运行结果与我们在客户端通过 db.startup_log.find()命令行获取到的结果是一致的。

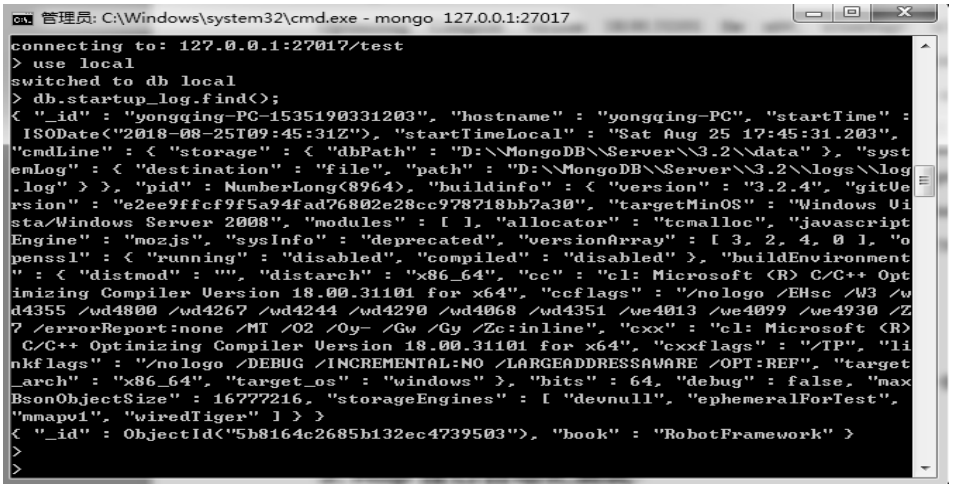

图 2-2-20

2.2.5 Update Many Mongodb Records

Update Many Mongodb Records 关键字用来更新 Collection 中的数据记录，接收[dbName | dbCollName | queryJSON | updateJSON | upsert=False]五个参数。

【示例】我们更新上面示例中插入的{"book":"RobotFramework"}记录为{"book":"robotFramework"}，即将 RobotFramework 变为 robotFramework，如图 2-2-21 所示。

Connect To MongoDB	127.0.0.1	27017	2	
${newJson}	**Set Variable**	{"$set": {"book":"robotFramework"}}		
Update Many Mongodb Records	local	startup_log	{"book":"RobotFramework"}	${newJson}
Disconnect From Mongodb				

图 2-2-21

运行结果如图 2-2-22 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase006
20180825 23:05:53.692 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
20180825 23:05:53.700 :  INFO : ${newJson} = {"$set": {"book":"robotFramework"}}
20180825 23:05:53.708 :  INFO :
Matched: 1 documents
| ${allResults} | Update Many MongoDB Records | local | startup_log | {u'book':
u'RobotFramework'} | {u'$set': {u'book': u'robotFramework'}} |
20180825 23:05:53.710 :  INFO : | Disconnect From MongoDB |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase006
```

图 2-2-22

更新完成后，通过客户端的 db.startup_log.find()命令来进行重新查询，如图 2-2-23 所示。从查询的结果可以看到指定的记录已经更新完成了。

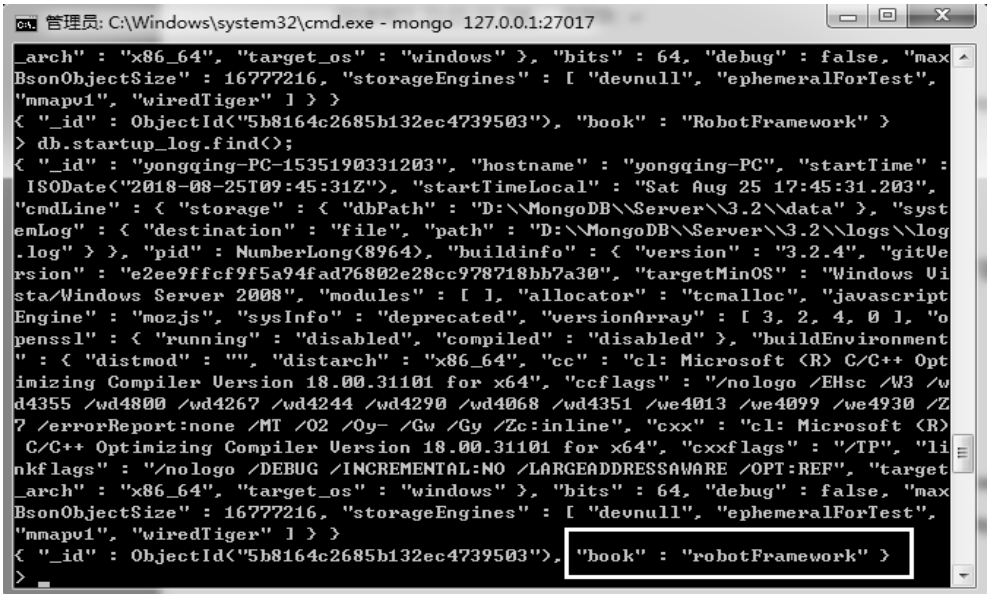

图 2-2-23

2.2.6　Remove Mongodb Records

Remove Mongodb Records 关键字用来删除指定 Collection 中的数据记录，接收[dbName | dbCollName | recordJSON]三个参数。

【示例】我们重新创建一个 capped 属性为 false 的 Collection，因为之前的 Collection 的 capped 属性为 true，会导致数据记录无法被删除。在客户端创建一个 Collection 的命令为 db.createCollection("RobotFramework",{capped : false})，如图 2-2-24 所示，新的名叫 RobotFramework 的 Collection 创建完成。

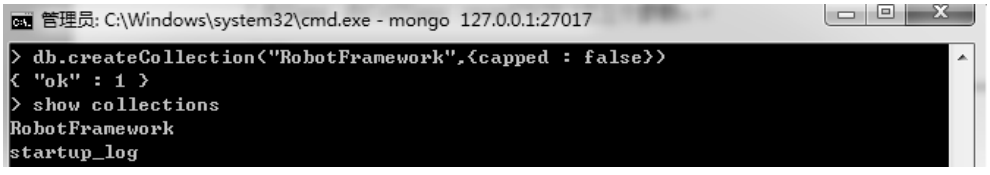

图 2-2-24

创建完成后，通过客户端的 db.Robot Framework.find()命令进行查询，如图 2-2-25 所示。然后我们使用 Remove Mongodb Records 关键字来删除{"book":"robotFramework"}这条记录，如图 2-2-26 所示。

```
> use local
switched to db local
> db.RobotFramework.find()
{ "_id" : ObjectId("5b8202ee685b130624a2bc4d"), "book" : "RobotFramework" }
>
```

图 2-2-25

Connect To MongoDB	127.0.0.1	27017	2
${Json}	**Set Variable**	{"book":"RobotFramework"}	
Remove Mongodb Records	local	RobotFramework	${Json}
Disconnect From Mongodb			

图 2-2-26

运行结果如图 2-2-27 所示。

```
Starting test: RobotFrameworkTest1.TestSute13.TestCase007
20180826 09:40:05.945 :  INFO :
| Connect To MondoDB | dbHost | dbPort | dbMaxPoolSize | dbNetworktimeout |
dbDocClass | dbTZAware |
| Connect To MondoDB | 127.0.0.1 | 27017 | 2 | None | <type 'dict'> | False |
20180826 09:40:05.947 :  INFO : ${Json} = {"book":"RobotFramework"}
20180826 09:40:05.970 :  INFO : | ${allResults} | Remove MongoDB Records | local
| RobotFramework | {u'book': u'RobotFramework'} |
20180826 09:40:05.972 :  INFO : | Disconnect From MongoDB |
Ending test:   RobotFrameworkTest1.TestSute13.TestCase007
```

图 2-2-27

执行完成后，在客户端执行 db.RobotFramework.find()进行查询，如图 2-2-28 所示，从查询的结果可以看到{"book":"robotFramework"}这条记录已经被删除。

图 2-2-28

2.2.7 MongoDBLibrary 库的其他关键字

表 2-2-1 中列出了 MongoDBLibrary 库中其他关键字的使用示例。

表 2-2-1　MongoDBLibrary 库中其他关键字

<table>
<tr><th>关键字</th><th>使用描述</th></tr>
<tr><td>Drop Mongodb Database</td><td>该关键字用来删除指定的 MongoDB 数据库，接收[dbDelName]一个参数，示例：
<table>
<tr><td>Drop Mongodb Database</td><td>local</td></tr>
</table></td></tr>
<tr><td>Drop Mongodb Collection</td><td>该关键字用来删除指定的 Collection，接收[dbName | dbCollName]两个参数，示例：
<table>
<tr><td>Drop Mongodb Collection</td><td>local</td><td>RobotFramework</td></tr>
</table></td></tr>
<tr><td>Get Mongodb Collection Count</td><td>该关键字用来获取指定 Collection 下的数据记录总数，接收[dbName | dbCollName]两个参数，示例：
<table>
<tr><td>${counts}</td><td>Get Mongodb Collection Count</td><td>local</td><td>RobotFramework</td></tr>
<tr><td>log</td><td>${counts}</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Retrieve And Update One Mongodb Record</td><td>该关键字用来获取并且更新指定的数据记录，接收[dbName | dbCollName | queryJSON | updateJSON | returnBeforeDocument=False]五个参数，示例：
<table>
<tr><td>${queryJson}</td><td>Set Variable</td><td>{"book":"RobotFramework"}</td><td></td><td></td></tr>
<tr><td>${newJson}</td><td>Set Variable</td><td>{"$set":{"book":"robotFramework"}}</td><td></td><td></td></tr>
<tr><td>Retrieve And Update One Mongodb Record</td><td>local</td><td>RobotFramework</td><td>${queryJson}</td><td>${newJson}</td></tr>
</table></td></tr>
<tr><td>Retrieve Mongodb Records With Desired Fields</td><td>该关键字用来从 Collection 中根据指定的字段查询出对应的满足要求的数据记录，接收[dbName | dbCollName | recordJSON | fields | return__id=True | returnDocuments=False]六个参数，示例：
<table>
<tr><td>${result}</td><td>Retrieve Mongodb Records With Desired Fields</td><td>local</td><td>RobotFramework</td><td>{}</td><td>book</td></tr>
<tr><td>log</td><td>${result}</td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Retrieve Some Mongodb Records</td><td>该关键字用来从 Collection 中查询出根据指定 JSON 匹配到的数据记录，接收[dbName | dbCollName | recordJSON | returnDocuments=False]四个参数，示例：
<table>
<tr><td>${result}</td><td>Retrieve Some Mongodb Records</td><td>local</td><td>RobotFramework</td><td>{"book":"robotFramework"}</td></tr>
<tr><td>log</td><td>${result}</td><td></td><td></td><td></td></tr>
</table></td></tr>
</table>