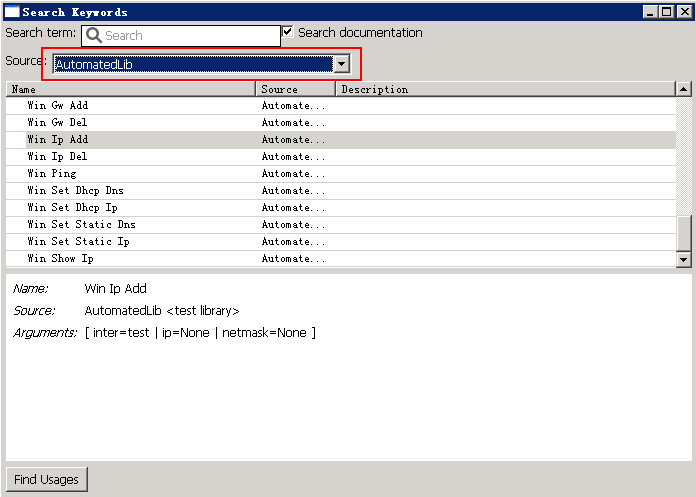
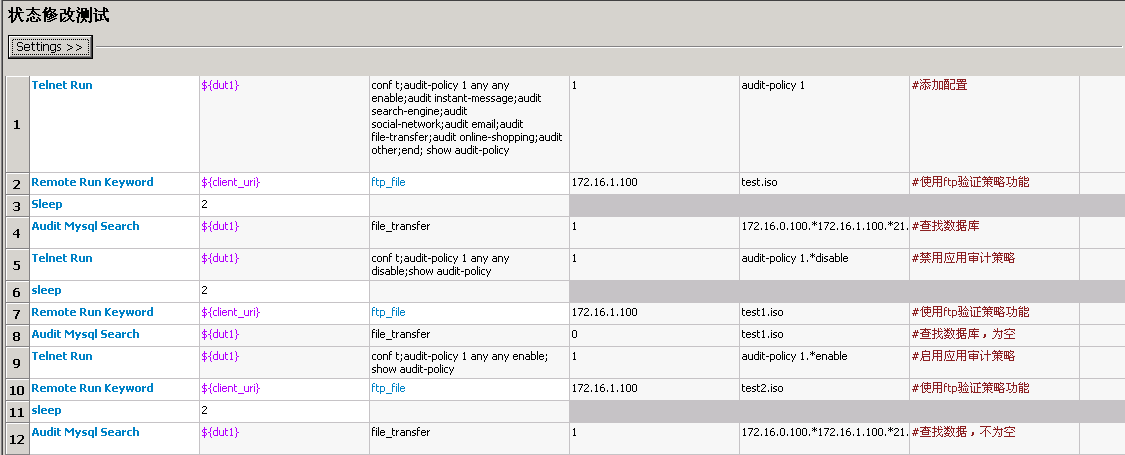
Robot Framework 关键字和远程库

# 系统关键字

## 目标

可以使用F5加载封装的关键字，并可在case中调用。





## 封装关键字

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

#-\*- coding:utf-8 -\*-

'''

created by bugmaster 2015-01-29

'''

\_\_version\_\_ = '0.1'

from robot.api import logger

import os

class Runbat(object):

def run\_all\_bat(self,path):

u'''接收一个目录的路径，并执行目录下的所有bat文件.例

| run all bat | filepath |

'''

for root,dirs,files in os.walk(path):

for f in files:

if os.path.splitext(f)[1] == '.bat':

os.chdir(root)

#print root,f

os.system(f)

def \_\_execute\_sql(self, path):

logger.debug("Executing : %s" % path)

print path

def decode(self,customerstr):

return customerstr.decode('utf-8')

if \_\_name\_\_ == "\_\_main\_\_":

path = u'D:\\test\_boject'

run = Runbat()

run.run\_all\_bat(path)

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

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

from runbat import Runbat

\_\_version\_\_ = '0.1'

class CustomLibrary(Runbat):

"""

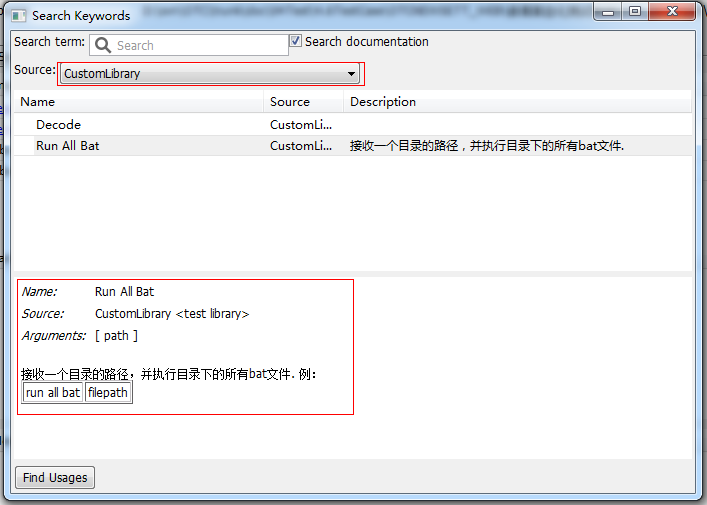
这里也可以装x的写上我们创建的CustomLibrary如何如何。

"""

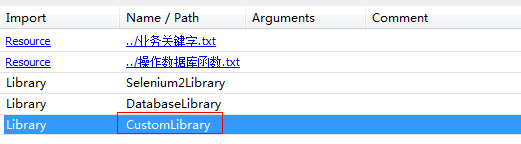
ROBOT\_LIBRARY\_SCOPE = 'GLOBAL'

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

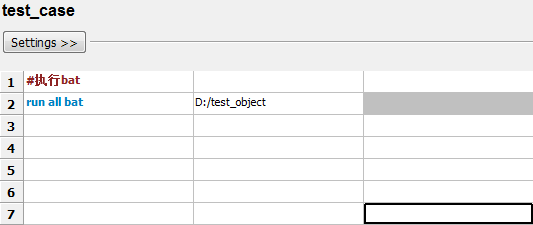
1. 下面，启动robot framework RIDE，按F5:



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



然后，在具体的测试用例中使用“run all bat” 关键字。

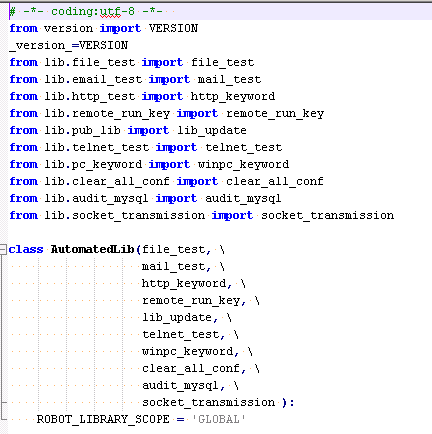


其实核心还是会点Python ，利用工具，但又不受制于工具。

## 当前系统的关键字及加载

1. 自动化库 C:\Python27\Lib\site-packages\AutomatedLib

AutomatedLib文件加的\_\_init\_\_.py内容如下：



from lib.aaa import bbb 从当前文件夹lib的aaa.py文件中，加载bbb类

AutomatedLib继承类。

**注意：所有自定义封装的关键字名称不能相同，否则会出现继承错误问题。**

1. 以下是lib/clear\_all\_conf.py的关键字：

# -\*- coding:utf-8 -\*-

from AutomatedLib.lib.pub\_lib import \_unicode\_to\_utf,\_searchchar,ExpectError

from AutomatedLib.configure import dut\_user,dut\_passwd

import telnetlib,re,time

class clear\_all\_conf(object):

def \_\_init\_\_(self):

pass

def \_login(self,ip,port,username,passwd):

tl=telnetlib.Telnet(ip,port,10)

tl.write(username+'\n')

time.sleep(1)

tl.write(passwd+'\n')

tl.write('enable\n')

tl.write('end\n')

return tl

def clear\_all\_conf(self,control\_ip,control\_port,dut\_mgt\_ip,user=dut\_user,passwd=dut\_passwd,t=190):

msgs=[]

cmds=['enable']

cmds.append('end')

cmds.append('erase startup-config ')

cmds.append('reboot')

tl=self.\_login(control\_ip,int(control\_port),user,passwd)

for cmd in cmds:

if cmd == 'reboot':

tl.write('%s\n' % cmd)

msg=tl.read\_until('y/n')

tl.write('y\n')

break

else:

tl.write('%s\n' % cmd)

msg=tl.read\_until('#')

msgs.append(msg)

tl.close()

time.sleep(t)

tl=self.\_login(control\_ip,int(control\_port),user,passwd)

cmds=[]

cmds.append('conf t')

cmds.append('interface mgt')

cmds.append('no ip address 192.168.1.200/24')

cmds.append('ip address '+dut\_mgt\_ip+'/16')

cmds.append('allow all')

cmds.append('end')

cmds.append('write memory')

for cmd in cmds:

tl.write('%s\n' % cmd)

msg=tl.read\_until('#')

msgs.append(msg)

print("%s"%' '.join(msgs))

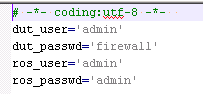
tl.close()

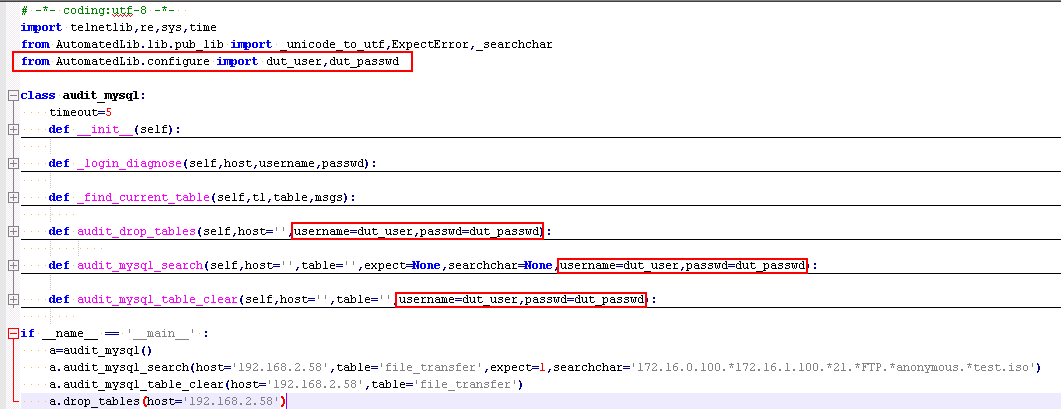
if \_\_name\_\_ == '\_\_main\_\_' :

test=clear\_all\_conf()

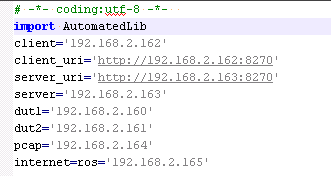
test.clear\_all\_conf('192.168.2.200','10002','192.168.2.58')

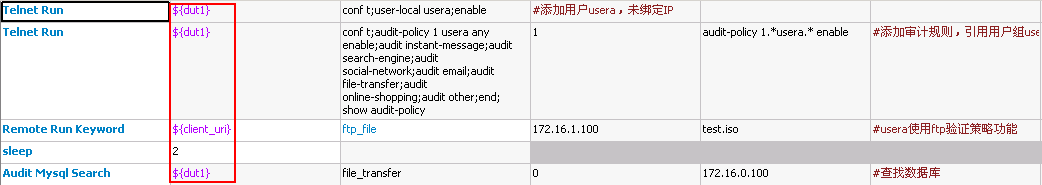
1. 变量
2. C:\Python27\Lib\site-packages\AutomatedLib\config.py 存放关键字引用的变量值。





1. 自动化case目录中也存在一个config.py 存放测试案例中的变量。





# 远程库

## 原理

1. **使用原因：**

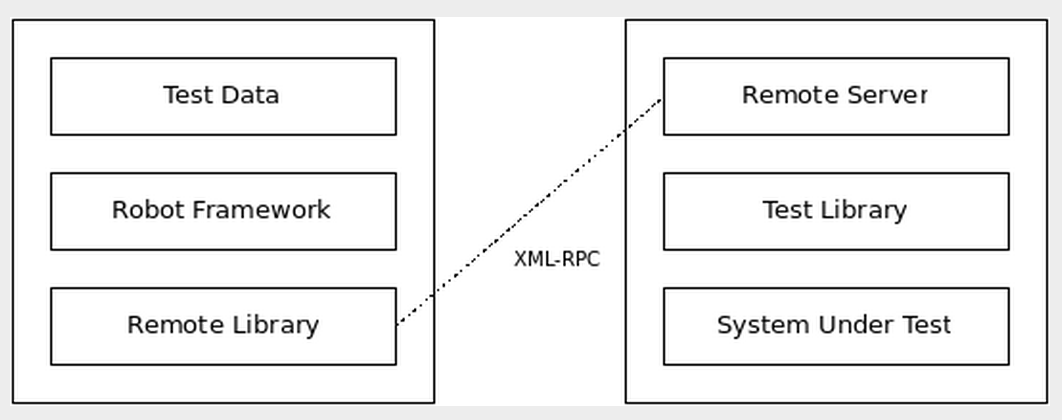
①测试需要跨越多个物理机器，且有的测试库也必须部署在被测系统上。比如：客户端需要在两台系统分别为windows和linux的服务器下，执行测试环境的初始化及清理工作。

②一个测试要使用多个库，但是有的只能用jybot运行，有的只能用pybot运行(这种情况很常见)。

1. **远程库接口的原理：**

为了解决上述两个难题，Robot Framework提供了远程库接口技术(remote library interface)。

什么是远程库接口技术呢?其实很简单，远程库接口就是把原来的测试库变成了三部分。一部分我们可以叫他远程库(Remote Library)，第二部分叫做远程[服务器](http://www.ltesting.net/ceshi/ruanjianceshikafajishu/rjcshjdj/wlfwq/)(Remote Server)，第三部分是真正的测试库(Test Library)。测试库提供真正的测试功能，它被远程[服务器](http://www.ltesting.net/ceshi/ruanjianceshikaifajishu/rjcshj)包裹起来，通过XML-RPC协议被远程库访问(见下图)。它的实现思路说白了就是设计模式中的Proxy模式。



**Test Data** ： 就是测试用例。

**RobotFramework** 不用讲啦，就是控制端。

**Remote Library** ：C:\Python27\Lib\site-packages\robot\libraries\Remote.py，用于连接远程服务器的。

**Remote Server**：远程服务器，用于发布本地测试关键字库。http://robotframework.googlecode.com/hg/tools/remoteserver/robotremoteserver.py把下载的robotremoteserver.py，放在一个python可以搜索到的目录（默认放到了AutomatedLib目录中）。

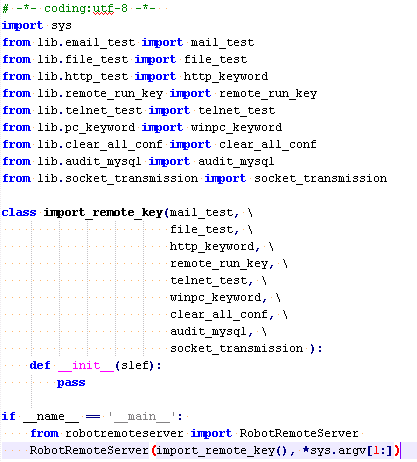
**Test Library：**本地关键字库。

**System Unter Test：**执行关键字。

## 远程服务

**Remote Server**：远程服务，用于发布本地测试关键字库。http://robotframework.googlecode.com/hg/tools/remoteserver/robotremoteserver.py把下载的robotremoteserver.py，放在一个python可以搜索到的目录（默认放到了AutomatedLib目录中）。然后创建一个Remote Server

**例如：**自动化环境中创建本地发布关键字服务Remotelibrary.py 添加代码如下：



这段代码先创建了一个测试库import\_remote\_key，并且继承自定义封装的关键字库。然后使用RobotRemoteServer()对象，创建了一个远程服务器，默认地址为"localhost",默认端口为"8270"。

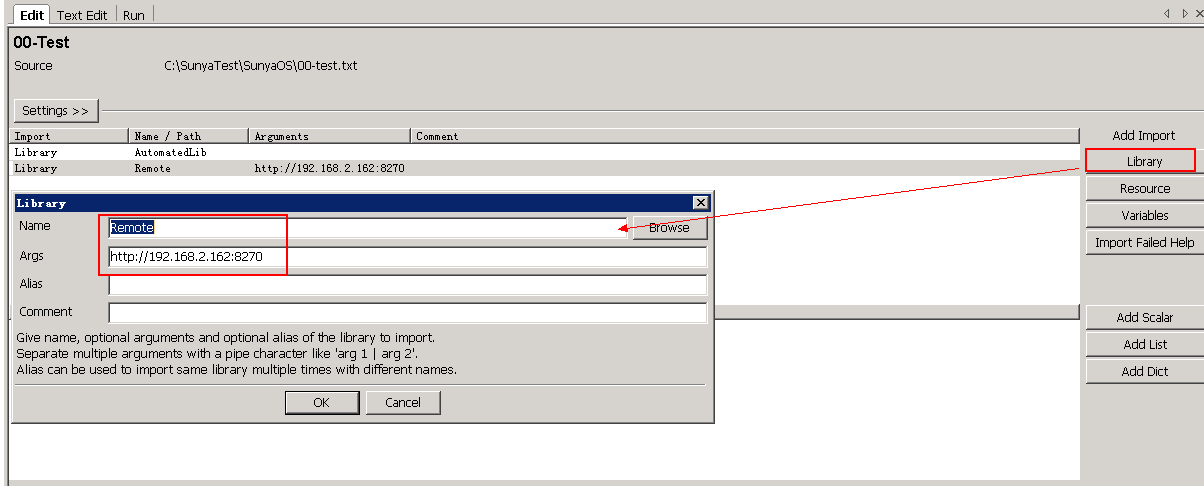
**注意，RobotRemoteServer对象的参数“import\_remote\_key()”，传入的是一个对象，而不是字符串。**

执行后显示如下：

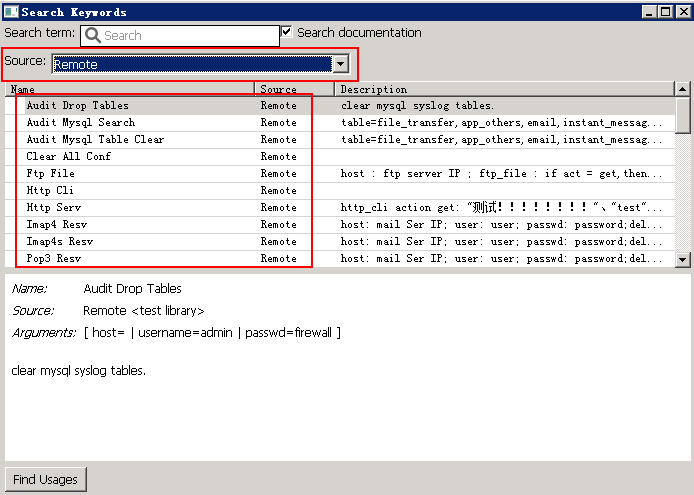


## 远程库

用于连接远程服务的，C:\Python27\Lib\site-packages\robot\libraries\ Remote.py，安装robot后自带的库。



按F5，查看远程服务器发布的关键字



使用远程关键字，和本地关键字一样，直接使用即可。

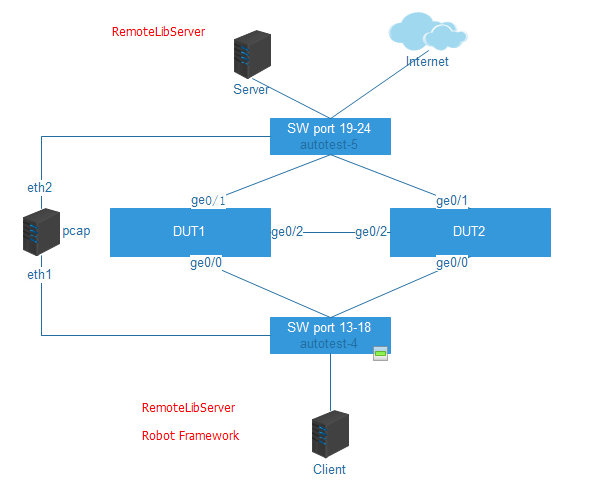
**注意本地库的关键字和远程服务器的关键字名称不要一样，如果一样，则使用库名.关键字加以区分。**



# 应用

## 原理和拓扑

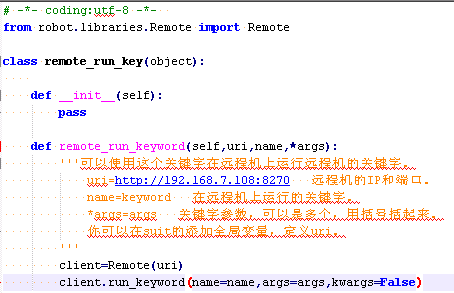
自动化框架中，使用了远程服务器框架，且本地关键字库和远程服务器关键字库是相同的。本地封装关键字Remote Run Keyword，用来执行远程服务器库中的关键字。一是为了防止编写出的自动化case混乱，二是robot Fremework进程的可移动性（不受RobotFramework位置的影响）。



自动化拓扑

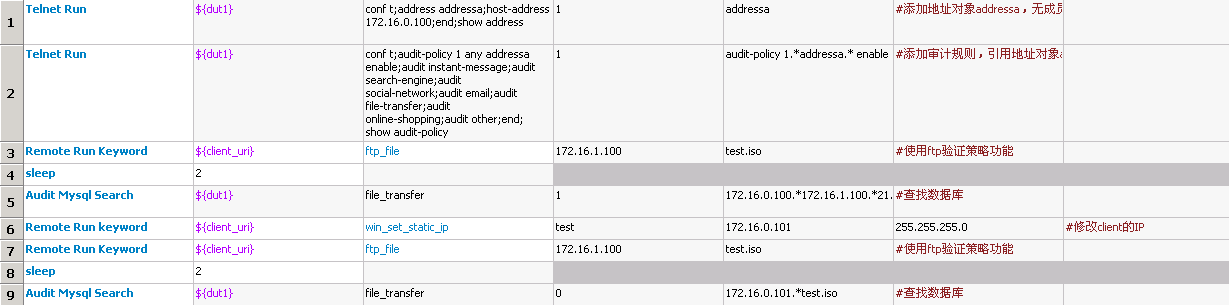
在Client和Server上分别发布自动化关键字库（远程服务器），在Robot Framework上执行关键字，RobotFramework会连接Client和Server，执行其上的关键字。

通过remote加载远程服务的关键字无法实现一个案例中交替操作多台设备，所以本地封装的remote\_run\_keywork（执行远程服务上关键字）关键字代码如下：



## case

示例：



Remote Run Keyword | ${client\_uri} | ftp\_file | 172.16.1.100 | test.iso #使用ftp验证策略功能

**Remote Run Keyword** 关键字

**${client\_uri}** 远程服务器地址，变量值在case的config.py中定义。

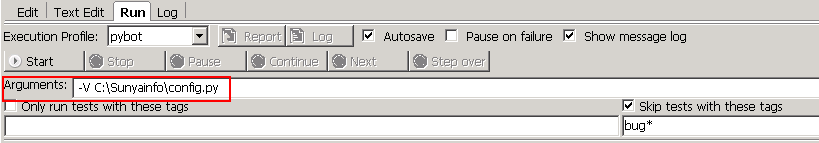
**ftp\_file** 远程服务器上的关键字

**172.16.1.100| test.iso** ftp\_file关键字的参数

## 执行

执行时，需要加载变量，否则无法找到case中的**${client\_uri}**的值，使用-V参数。

示例：



# 参考文档：

http://www.cnblogs.com/fnng/p/4261293.html

http://blog.csdn.net/liuheng123456/article/category/1315555