Shetty et al., J Comput Eng Inf Technol 2020, 9:3
DOI: 10.37532/jceit.2020.9(3).227

Journal of Computer Engineering & Information Technology

Research Article

A SCITECHNOL JOURNAL

An Empirical Study on Robot Test Automation Framework

Vijayashree Shetty*, Swetha S and Bindu Ashwini C

Abstract

Recently with the usage of robot framework there has been a convincing improvement in IT automation process, improving both quality and efficiency of products with the decreased association of Tester. Robot Framework is open and extensible. A ground-breaking and adaptable automation solution can be made by integrating Robot Framework with any other tool. It ensures easy extensibility and reusability through its ability to form high-level keywords from existing keywords. Further its abilities can be stretched out by libraries implemented with Python or Java. The framework incorporates a rich ecosystem around it, consisting of libraries and tools that are developed as separate projects, a command line interface and XML (Extensible Markup Language) output files simplify the integration into existing infrastructure. This paper describes the extensible characteristic set which makes Robot framework exceptionally qualified for automation of test cases in current development environment within short time frame. We further explain the usage of framework using a sample test suite with its log and reports, also exploring different available libraries and tools available.

Keywords: Cryptography; Encryption; Symmetric; Asymmetric; Robot framework; Selenium; Automation; Manual Testing; Keyword-Driven

Introduction

Automation Testing utilizes specialized testing tools and reduce the human or manual interruption, redundant or repetitive errands. Use of automation in testing saves both time and efforts [1]. Numerous automation tools are available that are commercial and open source, out of which some of them are sophisticated and use proprietary or existing coding languages. The style wherein test cases are written can be keyword driven, behavior driven and data driven. Keyword driven testing, test scripts are executed at higher level of abstraction. Robot framework is one such test automation framework which is technology independent for acceptance test-driven development (ATDD). The development of this tool was sponsored by Nokia Siemens Networks and was open sourced in 2008.

In Robot framework test cases are written in tabular format using keyword driven approach. Robot Framework goes well with all Operating Systems available. It was built on Python and runs on Python and Iron Python. The Framework has an extensible support for external tools and libraries which are open source, Selenium Library is one among them which aids interacting with the browsers and helps with web application and UI testing. Operating System Library, MongoDB Library, String Library, Android Library, Java Libcore are some of the other available libraries. In our test case we will be using Selenium Library to interact with Browser. This paper is organized as follows. Section 2 explains the architecture of framework. Section 3 describes libraries and tools provided by the framework. Section 4 describes the benefits of using framework. Section 5 describes the execution of test cases in the framework. Section 6 compares manual testing with automation using framework. Section 7 draws conclusion.

Architecture of Framework

Test execution is begun from the command line. Some of the IDE's like eclipse provide an option to execute the test case directly without using CL. At the point when execution is begun, the framework initially parses the information. It uses keywords furnished by the libraries, executes test cases. Interaction with target system under test is handled by test libraries. Libraries in turn may use lower level tool as drivers or use interfaces directly. The framework offers several standard libraries that which provides actual testing capabilities during testing by providing keywords [2]. It also automatically generates the report and logs for reference Figure 1.

Test data

Test data is in the tabular form. To begin with test execution framework parses these data to run test case

Robot framework

A test automation framework that requires python to be installed on your system to run the test cases.

Test Libraries

A number of standards, external and other libraries are made available to aid in automation by providing variety of keywords.

Test tool

Framework in addition to libraries also provide tools to facilitate

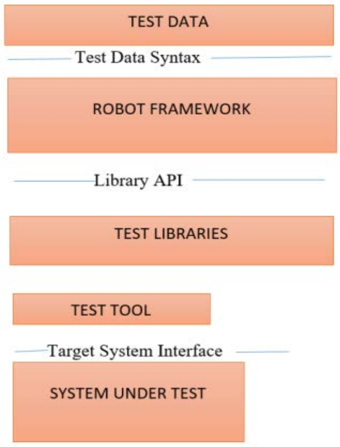


Figure 1: Architecture of Robot Framework.

***Corresponding author:** Vijayashree Shetty, Department of Information Science and Engineering, R V College of Engineering, Bengaluru. E-mail: vijayashrees.is16@rvce.edu.in

Received: May 18, 2020 **Accepted:** July 24, 2020 **Published:** July 31, 2020

SciTechnol International Publisher of Science, Technology and Medicine

All articles published in Journal of Computer Engineering & Information Technology are the property of SciTechnol, and is protected by copyright laws. Copyright © 2020, SciTechnol, All Rights Reserved.

other actions like generating report, keyword documentations, clean up robot files.

System under automation testing

Represents actual system that is being tested, with whom framework interacts.

Libraries and tools

Libraries give the real automation and testing capacities to Robot Framework by giving keywords. A few standard libraries are packaged with the framework, and there are independently evolved external libraries in abundance that can be installed depending on your requirements.

Built In is a Standard Library imported naturally and, in this way, constantly accessible. It that gives a lot of nonexclusive keywords required regularly for instance, or conversions(e.g Convert to Binary, convert to Boolean) for verifications (e.g. Should Be Empty, Should Be Equal, Should Contain) and for different purposes (e.g. Sleep, Log to Console, Run Keywords).

Operating System is a Standard Library that provides keywords that enable various Operating System related tasks where the framework is running. (e.g. Set Environment Variable, Run, Create File ,File Should Exist, Copy Files)

A web testing Library that uses the Selenium WebDriver modules internally to control the elements on the web browser. For the keywords to interact with the web elements needs a locator argument to be passed to it while using. Locator can be an id of the web element, css property or an Xpath. Keywords provided by the framework includes Click Element, Click Button, Drag and Drop, Get Cookies, Input Text, Press Key etc.

External Library for interacting with MongoDB using pymongo(e.g. Connect To Mongodb,Get Mongodb Databases, Retrieve All Mongodb Records, Drop Mongodb Database).

Builtin tool for generating logs and reports. One among its important feature is that the ability to merge the output from various other test execution rounds. This helps in generating an overall report of running the identical test cases on different environments.

Test Data Editor offered by the framework. Provides a staright forward way of writing test suites for automation through handy keyboard shortcuts.

Framework provides plugin for eclipse IDE.It is useful not only for writing testcases but also to look at the source of libraries and writing new libraries. When it comes to maintaining test cases both RIDE and Eclipse plugin are equally good. We use Eclipse plugin in our example test case.

Benefits of Using Robot Framework

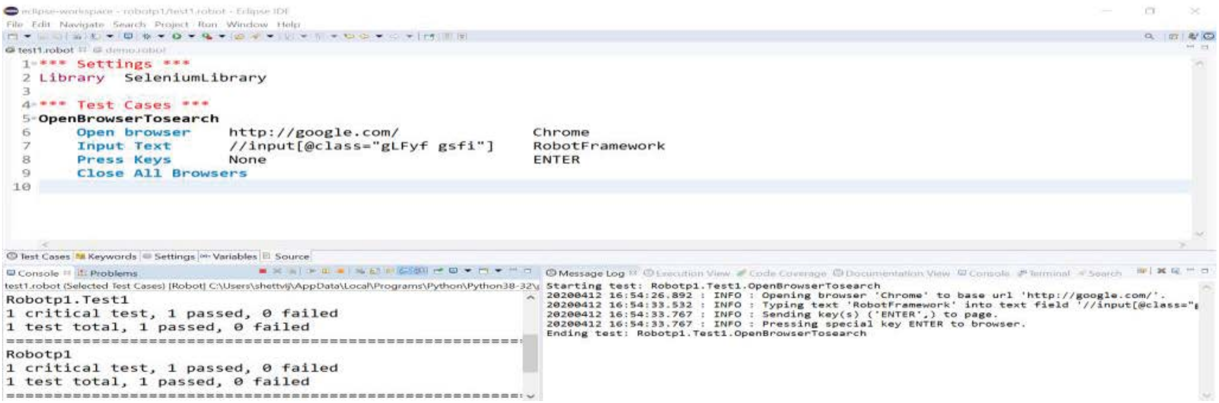
Manual testing cannot explore the whole product effectively, debugging becomes even more difficult and time consuming with manual efforts. Whereas with frameworks like robot, the whole process just needs an individual to execute a command and check results later on. Tester can be engaged with some other work. Also, the way in which framework is designed to write testcases follow a natural flow of precondition, action, verification and cleanup [3]. RIDE (Robot Integrated Development Environment) is a graphical user interface is provided by the framework that aids in writing the testcases. Descriptions for each of the keyword is provided in such way that even a non-technical person can understand the meaning. All the reports and log information are generated automatically after the execution of command. The cost of automating a test is best measured by the number of manual tests prevented from running and the bugs it will therefore cause to miss [4], and this is probably the biggest strength of the Robot Framework.

Example of Testcase Execution Using Eclipse Ide

In order to mitigate manual testing efforts, test scripts are written with the help of keywords that are simple and consumes less time, hence ease out most of the testing effort. In Robot Framework, a test case is basically a combination of keywords that perform some defined set of tasks. We can either use a built-in keyword or can define our own keyword. Our example uses Selenium Library keywords to open chrome browser and search about Robot Framework as shown in Figure 2.

Test case can be executed from the terminal, which leads to generation of execution report by the framework. It can also be executed from IDE directly. All the critical test cases should be passed in order for a successful completion of test. Once test cases are executed three types of file gets generated. Location/Links to the file will be available on the terminal. Background color of the report/log files shows the status of the executed test suite (pass/fail) as shown in Figures 3 and 4.

In addition to report file two other files are created, Output.Xml



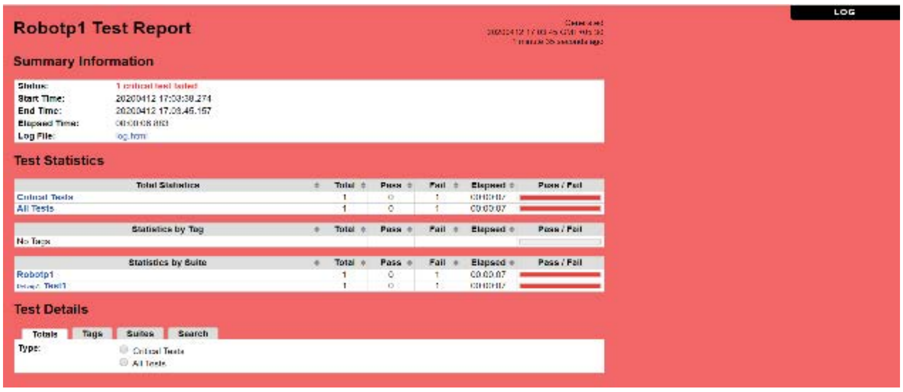
```
*** Settings ***
Library   SeleniumLibrary

*** Test Cases ***
OpenBrowserTosearch
    Open browser     http://google.com/                 Chrome
    Input Text       //input[@class="gLFyf gsfi"]       RobotFramework
    Press Keys       None                               ENTER
    Close All Browsers
```

```
test1.robot (Selected Test Cases) [Robot] C:\Users\shettvij\AppData\Local\Programs\Python\Python38-32\
Robotp1.Test1
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================
Robotp1
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================
```

```
Starting test: Robotp1.Test1.OpenBrowserTosearch
20200412 16:54:26.892 : INFO : Opening browser 'Chrome' to base url 'http://google.com/'.
20200412 16:54:33.532 : INFO : Typing text 'RobotFramework' into text field '//input[@class="g
20200412 16:54:33.767 : INFO : Sending key(s) ('ENTER',) to page.
20200412 16:54:33.767 : INFO : Pressing special key ENTER to browser.
Ending test: Robotp1.Test1.OpenBrowserTosearch
```

Figure 2: Test case example.

doi: 10.37532/jceit.2020.9(3).227



LOG

Robotp1 Test Report

Generated [illegible]

Summary Information

Status:	1 critical test failed
Start Time:	20200412 17:03:38.274
End Time:	20200412 17.03.46.157
Elapsed Time:	00:00:08.883
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	0	1	00:00:07	🟥
All Tests	1	0	1	00:00:07	🟥

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

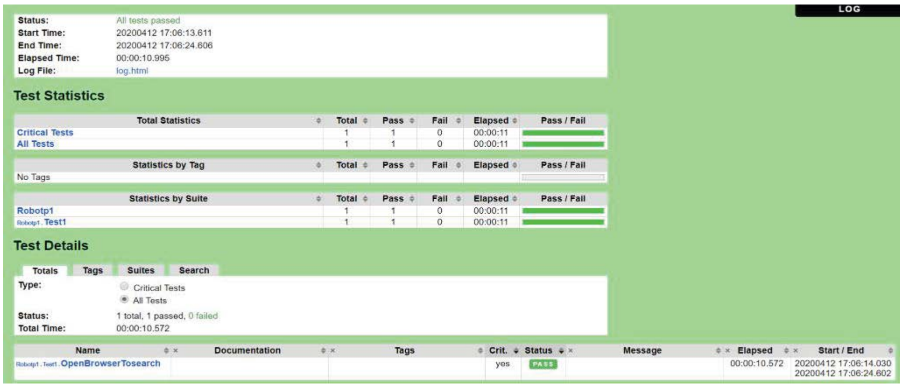
Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Robotp1	1	0	1	00:00:07	🟥
Robotp1 . Test1	1	0	1	00:00:07	🟥

Test Details

Totals | Tags | Suites | Search

Type: ○ Critical Tests ○ All Tests

Figure 3: Test case execution Report (Fail).



LOG

Status:	All tests passed
Start Time:	20200412 17:06:13.611
End Time:	20200412 17:06:24.606
Elapsed Time:	00:00:10.995
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:11	🟩
All Tests	1	1	0	00:00:11	🟩

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

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Robotp1	1	1	0	00:00:11	🟩
Robotp1 . Test1	1	1	0	00:00:11	🟩

Test Details

Totals | Tags | Suites | Search

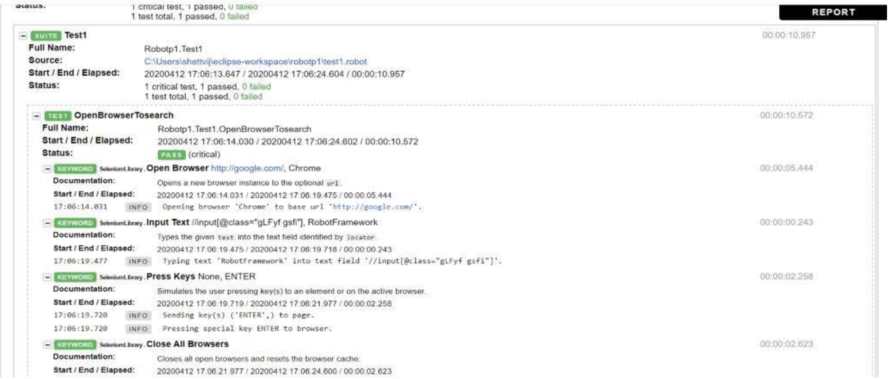
Type: ○ Critical Tests ● All Tests

Status: 1 total, 1 passed, 0 failed

Total Time: 00:00:10.572

Name	Documentation	Tags	Crit.	Status	Message	Elapsed	Start / End
Robotp1 . Test1 . OpenBrowserTosearch			yes	PASS		00:00:10.572	20200412 17:06:14.030 20200412 17:06:24.602

Figure 4: Test case execution Report (Pass).



REPORT

```
Status:                  1 critical test, 1 passed, 0 failed
                         1 test total, 1 passed, 0 failed

− SUITE Test1                                                                        00:00:10.957
  Full Name:             Robotp1.Test1
  Source:                C:\Users\shettvij\eclipse-workspace\robotp1\test1.robot
  Start / End / Elapsed: 20200412 17:06:13.647 / 20200412 17:06:24.604 / 00:00:10.957
  Status:                1 critical test, 1 passed, 0 failed
                         1 test total, 1 passed, 0 failed

  − TEST OpenBrowserTosearch                                                         00:00:10.572
    Full Name:             Robotp1.Test1.OpenBrowserTosearch
    Start / End / Elapsed: 20200412 17:06:14.030 / 20200412 17:06:24.602 / 00:00:10.572
    Status:                PASS (critical)
    − KEYWORD SeleniumLibrary.Open Browser http://google.com/, Chrome                00:00:05.444
      Documentation:         Opens a new browser instance to the optional url.
      Start / End / Elapsed: 20200412 17:06:14.031 / 20200412 17:06:19.475 / 00:00:05.444
      17:06:14.031   INFO   Opening browser 'Chrome' to base url 'http://google.com/'.
    − KEYWORD SeleniumLibrary.Input Text //input[@class="gLFyf gsfi"], RobotFramework  00:00:00.243
      Documentation:         Types the given text into the text field identified by locator.
      Start / End / Elapsed: 20200412 17:06:19.475 / 20200412 17:06:19.718 / 00:00:00.243
      17:06:19.477   INFO   Typing text 'RobotFramework' into text field '//input[@class="gLFyf gsfi"]'.
    − KEYWORD SeleniumLibrary.Press Keys None, ENTER                                 00:00:02.258
      Documentation:         Simulates the user pressing key(s) to an element or on the active browser.
      Start / End / Elapsed: 20200412 17:06:19.719 / 20200412 17:06:21.977 / 00:00:02.258
      17:06:19.720   INFO   Sending key(s) ('ENTER',) to page.
      17:06:19.720   INFO   Pressing special key ENTER to browser.
    − KEYWORD SeleniumLibrary.Close All Browsers                                     00:00:02.623
      Documentation:         Closes all open browsers and resets the browser cache.
      Start / End / Elapsed: 20200412 17:06:21.977 / 20200412 17:06:24.600 / 00:00:02.623
```

Figure 5: Log File created of executed of test case.

and a Log file which helps in further analysis.Log file gives detailed explanation of the tasks that got executed during the test as shown in Figures 5 and 6.

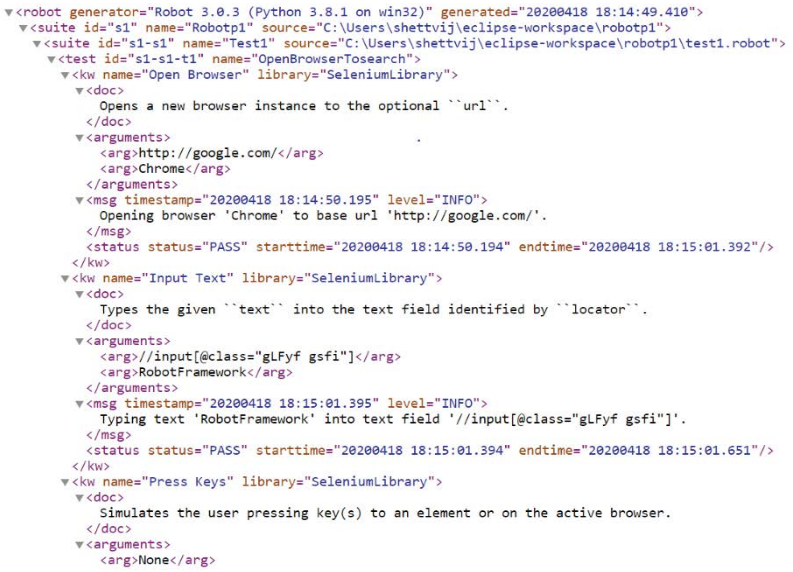
Manual testing and automation

It is important to know when and what type of activities need to be automated and not to be done manually [5] in testing environment. Software testing [6] suggests the usage of automatic or manual means to run a test on the software product. Conventional testing process that requires human interruption is tedious, non-specialized and wastes useful development time. Automation test [7-9] utilizes techniques, apparatuses and yield, decreasing manual mediation and accomplish unmanned gatekeeper finishing test, and naturally create logs and report of the test actions, which helps in examination of test after effects of test activities.

Clearly, the test automated innovation improves the productivity in testing and lessen repetitive work.

Conclusion

Robot Framework thus has a natural flow for automation and use keywords makes it easy for a non-technical person to understand the process and explore more. Ease of extension makes it a great



```
▼<robot generator="Robot 3.0.3 (Python 3.8.1 on win32)" generated="20200418 18:14:49.410">
  ▼<suite id="s1" name="Robotp1" source="C:\Users\shettvij\eclipse-workspace\robotp1">
    ▼<suite id="s1-s1" name="Test1" source="C:\Users\shettvij\eclipse-workspace\robotp1\test1.robot">
      ▼<test id="s1-s1-t1" name="OpenBrowserTosearch">
        ▼<kw name="Open Browser" library="SeleniumLibrary">
          ▼<doc>
             Opens a new browser instance to the optional ``url``.
           </doc>
          ▼<arguments>
             <arg>http://google.com/</arg>
             <arg>Chrome</arg>
           </arguments>
          ▼<msg timestamp="20200418 18:14:50.195" level="INFO">
             Opening browser 'Chrome' to base url 'http://google.com/'.
           </msg>
           <status status="PASS" starttime="20200418 18:14:50.194" endtime="20200418 18:15:01.392"/>
         </kw>
        ▼<kw name="Input Text" library="SeleniumLibrary">
          ▼<doc>
             Types the given ``text`` into the text field identified by ``locator``.
           </doc>
          ▼<arguments>
             <arg>//input[@class="gLFyf gsfi"]</arg>
             <arg>RobotFramework</arg>
           </arguments>
          ▼<msg timestamp="20200418 18:15:01.395" level="INFO">
             Typing text 'RobotFramework' into text field '//input[@class="gLFyf gsfi"]'.
           </msg>
           <status status="PASS" starttime="20200418 18:15:01.394" endtime="20200418 18:15:01.651"/>
         </kw>
        ▼<kw name="Press Keys" library="SeleniumLibrary">
          ▼<doc>
             Simulates the user pressing key(s) to an element or on the active browser.
           </doc>
          ▼<arguments>
             <arg>None</arg>
```

Figure 6: Output Xml file generated.

tool for use in IT automation process. Everything including testing, report generation, log file generation takes place automatically thus saving time and effort in continuous integration systems. Its multitudinous benefits make it highly eligible automation tool in current development process.

In the period of regular refreshing rendition of programming, repeatability of task irritates the test staff. The valuable test automation process improves the test process usage rate, decrease the test staff dreary activity, yet additionally the computerization test procedure can be performed without wasting much of tester's time. Also, testing work force can utilize these times to perform other useful activities improving productivity.

References

1. Kumar D, Mishra K, (2016) "The Impacts of Test Automation on Software's Cost, Quality and Time to Market"Procedia Computer Science,79: 8-15.
2. Stanislav S, Siemens C, Zelijko H (2011) "Usage of Robot Framework in Autmation of Functional Test Regression" Published in ICSEA Computer Science.
3. Garousi V, Mantyla MV (2016) "When and what to automate in software testing? A multi-vocal literature review". Information and Software Technology.
4. Jian-Ping L, Juan-Juan L, Dong-Long W (2012) "Application Analysis of Automated Testing Framework Based on Robot" Third International Conference on Networking and Distributed Computing,2: 21-24.
5. Eberhardinger B, Habermaier A, Reif W (2017) "Toward Adaptive, Self-Aware Test Automation." IEEE/ACM 12th International Workshop on Automation of Software Testing (AST).
6. Wu J, Chen C, Xiao L (2008) Software testing technology base. Huazhong university of science and technology press 5: 10-15.
7. Louise T, Bao X Wang X, Zhu G (2004) Software Testing. Posts & Telecom press, 20- 26.
8. Na Q, Huaichang D (2015) "Extension and application based on robot testing framework". 6th IEEE International Conference on Software Engineering and Service Science (ICSESS).
9. Marick B (1998) "When Should a Test Be Automated Proc.11th Intel Software/ Internet Quality Week.

Author Affiliation

Top

Department of Information Science and Engineering, R V College of Engineering, Bengaluru