Introduction to Robotframework

Kenneth Bellock

June 10, 2018

http://robotframework.org

Table of contents

Intended Audience

The intended audience for this presentation is someone with:

- No experience with robotframework, or an existing friend of the robot looking to learn a few new tricks or good practices
- A basic understanding of python

Objectives

This introduction is designed to start from nothing and to then incrementally build up concepts and capabilities to demonstrate an example of how robotframework tests can be organized and executed

Upon completion of examples in this presentation, you should be able to

- Install robotframework
- Say hello to the robot
- Group tests into suites and suites of suites
- Set timeouts for tests
- Write your own keywords
- Utilize variable files
- Create your own libraries of keywords from python
- Render pdf test reports

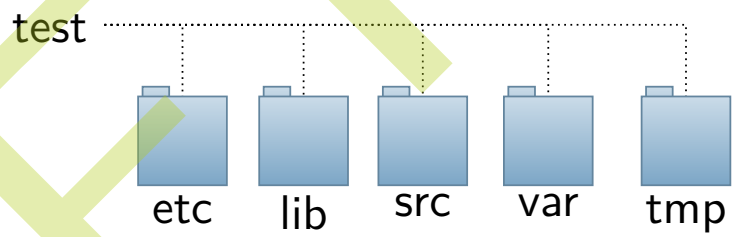
What is Robotframework?

From their webpage:

Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD). It has easy-to-use tabular test data syntax and it utilizes the keyword-driven testing approach. Its testing capabilities can be extended by test libraries implemented either with Python or Java, and users can create new higher-level keywords from existing ones using the same syntax that is used for creating test cases.

Directory Organization

- There is no *"Way of the Robot"* when it comes to organizing your project
- The framework is designed to be flexible, so you can define any organization you want, but this causes a lot of trouble for beginners and often results in projects where everything is just dumped into one big pot
- For these examples, we will use the Filesystem Hierarchy Standard to organize our components as shown to the right (`https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard`)



etc Resource Files

lib Libraries

src Test Suites

var Variables

tmp Temporary Files

Installation

- Virtualenv (`https://virualenv.pypa.io`), is a tool to create isolated Python environments; as demonstrated in Listing 1, it will provide the capability to install python toolboxes without administrator priveledges
- For these examples, I will use the locally qualified path to a virtual environment we are going to create
- If you already have an installation of robotframework available, substitute the path to your installation

Listing 1: Install Robotframework

```
1 >> virtualenv local
2 >> local/bin/pip install robotframework
```

Overview

Hello Robot

Lets start with the simplest thing to ask the robot to do, nothing. . .

Objectives

- Say hello to the robot
  - Write our first test file
  - Introduce suites and tests
  - Execute robotframework
  - Examine the different types and purposes of the output files

Setup

- Create a test folder for this project called `01-HelloRobot`
- Place the contents of Listing 2 into a file called `Hello.robot` in a subdirectory named `src`

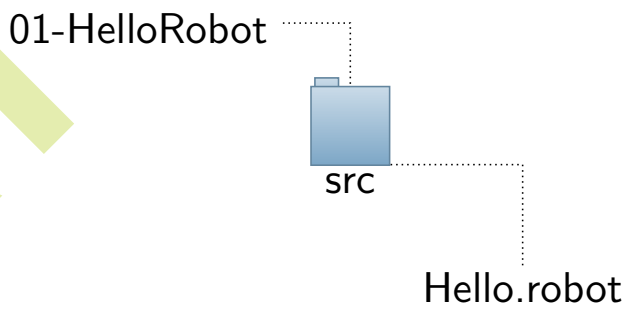
Listing 2: 01-HelloRobot/src/Hello.robot

```
1 *** Test Cases ***
2 | Hello Robot |
3 |   | No Operation |
```

- The file `Hello.robot` defines the test suite `Hello`
- Line 1 declares the start of our test cases for this suite
- Line 2 declares the start of a test named `Hello Robot`
- Line 3 is a step in this test, which is a builtin keyword that does nothing, it is appropriately named `No Operation`

Project Layout

- There is only one file, for projects this simple it is usually fine to place everything in one top level directory
- Since the following projects are going to build from this one, we are going to start with a more scaleable organization



Execution

From our project folder `01-HelloRobot`, execute the command in Listing 3

Listing 3: Say Hello

```
>> local/bin/robot src/Hello.robot
==============================================================================
Hello
==============================================================================
Hello Robot                                                           | PASS |
------------------------------------------------------------------------------
Hello                                                                 | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  output.xml
Log:     log.html
Report:  report.html
```

Execution started by processing the `Hello` suite, this consisted of the one test `Hello Robot`, then rolled up overall pass/fail and statistics for the suite, and ended with a list of output files

Output

- The primary output from a run is the `output.xml` file
  - This contains all the information logged by robotframework for a run
- The `log.html` and `report.html` are rendered from the `output.xml` file to provide quick views into the test results and execution steps
- The `output.xml` file is commonly passed to continuous integration servers like Jenkins or Buildbot to provide test summaries or to collect statistics
- If a different type of output file is desired, you can roll your own template emitter that parses the `output.xml` file, or you can write a listener (to be discussed in a later example)

Log

REPORT

Hello Test Log

Generated
20180609 20:25:17 GMT-04:00
11 hours 1 minute ago

Test Statistics

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

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

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Hello	1	1	0	00:00:00	

Test Execution Log

SUITE **Hello** 00:00:00.033

Full Name: Hello
Source: /local/bellockk/Development/IntroductionToRobotFramework/example/01-HelloRobot/src/Hello.robot
Start / End / Elapsed: 20180609 20:25:17.398 / 20180609 20:25:17.431 / 00:00:00.033
Status: 1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

TEST **Hello Robot** 00:00:00.001

Full Name: Hello.Hello Robot
Start / End / Elapsed: 20180609 20:25:17.430 / 20180609 20:25:17.431 / 00:00:00.001
Status: PASS (critical)

KEYWORD BuiltIn. **No Operation** 00:00:00.001

Documentation: Does absolutely nothing.
Start / End / Elapsed: 20180609 20:25:17.430 / 20180609 20:25:17.431 / 00:00:00.001

Report

LOG

Hello Test Report

Generated
20180609 20:25:17 GMT-04:00
11 hours 4 minutes ago

Summary Information

Status:	All tests passed
Start Time:	20180609 20:25:17.398
End Time:	20180609 20:25:17.431
Elapsed Time:	00:00:00.033
Log File:	log.html

Test Statistics

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

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

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Hello	1	1	0	00:00:00	

Test Details

Totals | Tags | Suites | Search

Type: ○ Critical Tests ○ All Tests

Summary

- With just three lines of test script, we were able to do a meet and greet with our new robot friend
- Robotframework, while being asked to do nothing, has done a lot of work
  - Meticulously logged each step of execution
  - Communicated with us while working
  - Collected statistics
  - Rendered html templates to provide log and results summaries

And I can do so much more!

Overview

Now that introductions are out of the way, lets create a real system under test (SUT), and see how we can leverage our new friend to run some tests on it

Objectives

- Execute a system command
- Log messages
- Relative pathing

Conclusion

- If this presentation worked well for you, please pass it on
- Send questions/comments to ken@bellock.net
- Contribute in development of this presentation at the github.com site below

`https://github.com/bellockk/IntroductionToRobotframework`