# Testing using Robot framework and xUnit reports

## Overview

In this tutorial, we will execute some tests using the Robot Framework.

> **Please note**
>
> This tutorial explores the integration using the JUnit XML report that the Robot Framework is capable of generate.
>
> However, the Robot Framework native XML format is supported by Xray and thus it should be the preferable way of importing tests/results from Robot test cases. Whenever integrating through Robot's specfic XML format, you have access to more features than the ones that are available if you use JUnit's XML format, since the latter is a generic format.

## Requirements

- robot framework
- Java (if using the Java variant of the "robot framework")

## Description

This and more examples may be found in Robot framework's robotdemo repository.

**calculator.py**

```
class Calculator(object):
    BUTTONS = '1234567890+-*/C='
    def __init__(self):
        self._expression = ''
    def push(self, button):
        if button not in self.BUTTONS:
            raise CalculationError("Invalid button '%s'." % button)
        if button == '=':
            self._expression = self._calculate(self._expression)
        elif button == 'C':
            self._expression = ''
        elif button == '/':
            self._expression += '//'    # Integer division also in Python 3
        else:
            self._expression += button
        return self._expression
    def _calculate(self, expression):
        try:
            return str(eval(expression))
        except SyntaxError:
            raise CalculationError('Invalid expression.')
        except ZeroDivisionError:
            raise CalculationError('Division by zero.')

class CalculationError(Exception):
    pass
```

Next, follows the robot test.

**keyword_driven.robot**

```
*** Settings ***
Documentation     Example test cases using the keyword-driven testing approach.
...
...               All tests contain a workflow constructed from keywords in
...               ``CalculatorLibrary.py``. Creating new tests or editing
...               existing is easy even for people without programming skills.
...
...               The _keyword-driven_ appoach works well for normal test
...               automation, but the _gherkin_ style might be even better
...               if also business people need to understand tests. If the
...               same workflow needs to repeated multiple times, it is best
...               to use to the _data-driven_ approach.
Library           CalculatorLibrary.py
*** Test Cases ***
Push button
    Push button    1
    Result should be    1
Push multiple buttons
    Push button    1
    Push button    2
    Result should be    12
Simple calculation
    Push button    1
    Push button    +
    Push button    2
    Push button    =
    Result should be    3
Longer calculation
    Push buttons    5 + 4 - 3 * 2 / 1 =
    Result should be    3
Clear
    Push button    1
    Push button    C
    Result should be    ${EMPTY}    # ${EMPTY} is a built-in variable
```

After running the tests (see bellow) and generating the JUnit XML report (e.g. robot.xml), it can be imported to Xray (either by the REST API or through "Import Execution Results" action within the Test Execution).

If you're using Python,

```
robot -x robot.xml keyword_driven.robot
```

Or if you're using Java,

```
java -jar robotframework-3.0.jar   -x robot.xml  keyword_driven.robot
```

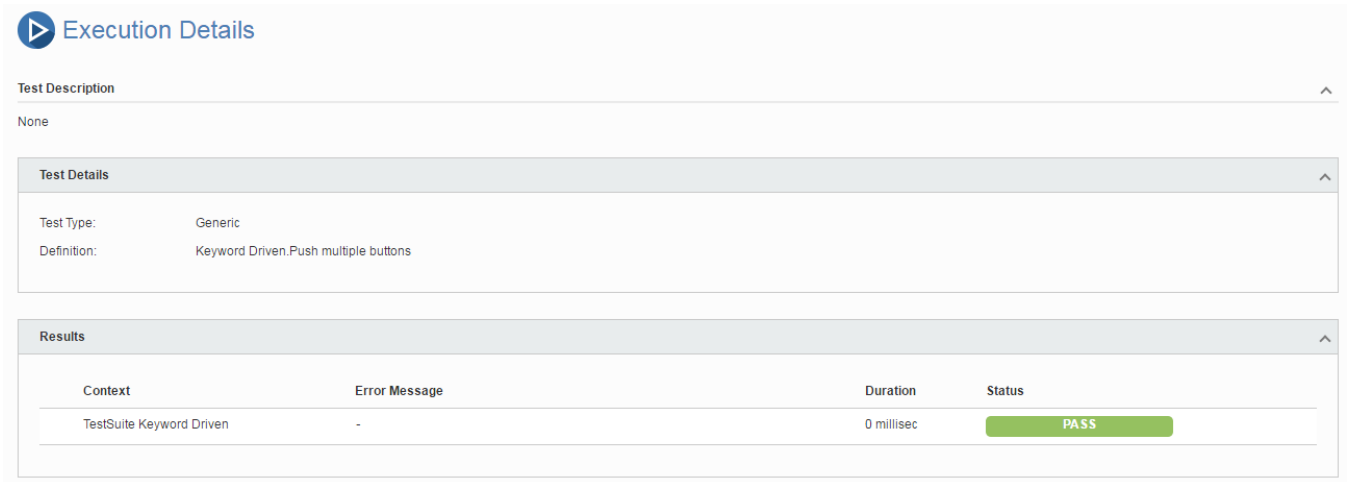
JUnit's Test Case is mapped to a Generic Test in JIRA, and the "Generic Test Definition" field contains the friendly name of the robot script concatenated with the aliias of the test case.

| | Key | Summary | Test Type | #Req | #Def | Test Sets | Assignee | Status | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | CALC-128 | Push multiple buttons | Generic | 0 | 0 | | | PASS | |
| 2 | CALC-127 | Push button | Generic | 0 | 0 | | | PASS | |
| 3 | CALC-129 | Simple calculation | Generic | 0 | 0 | | | PASS | |
| 4 | CALC-131 | Clear | Generic | 0 | 0 | | | PASS | |
| 5 | CALC-130 | Longer calculation | Generic | 0 | 0 | | | PASS | |

Showing 1 to 5 of 5 entries

First Previous 1 Next Last

The Execution Details of the Generic Test contains information about the Test Suite, which in this case corresponds to the friendly name of the robot script (i.e. "Keyword Driven").




# References

- http://robotframework.org/
- https://bitbucket.org/robotframework/robotdemo/src
- https://github.com/robotframework/robotframework/blob/master/INSTALL.rst