Taking advantage of Robot XML reports

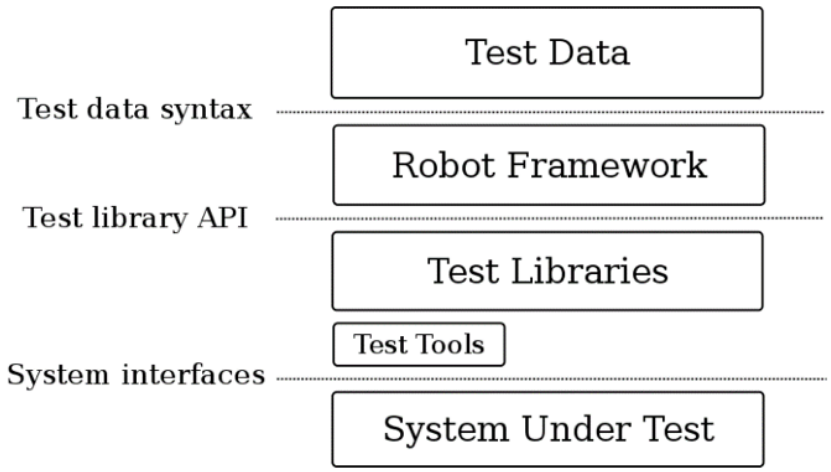
About the Robot Framework

The 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 uses the keyword-driven testing approach.

Robot Framework Concepts

The Robot Framework modular architecture can be extended with bundled and self-made test libraries.



A test suite consists of a file containing test cases. A directory containing these suite files creates a nested structure of test suites.

Robot is capable of generating a report for each execution in its specific XML format (xUnit) as well as HTML data formats.

The native Robot Framework's XML format has a well-defined schema and represents the pattern in which the framework executes the keywords given in that particular test suite.

Importing Robot Framework XML reports

Each Robot test case can be marked with a Tag identifying the requirement being tested; it also can be a Test definition already present in Jira. Tagging a requirement in a test case will create a link between the test case and the requirement. Identifying a test case with an issue key Tag, will create a test execution for the pre-existing Test (it does not create any new Test in this case).

It is important that the "Requirements issue types mapping" is configured at the point of import; otherwise, no link will be created. Refer to Issue Type Mapping for more information.

Below is a simplified example of a Robot Framework XML report. There are some nested test suites, and one failed test marked with Tags. The WEB-1 tag corresponds to an existing Story we want to test, while WEB-3 is the Test issue key which contains the test definition.

Robot-Framework example Suite and Test Case source file

```
*** Settings ***
Documentation     A test suite with a single Gherkin style test.
Library         BuiltIn
Library         Selenium2Library
Test Teardown     Close Browser

*** Variables ***
${SERVER}         localhost:8080
${BROWSER}        chrome
${DELAY}          0.5
${LOGIN URL}      http://${SERVER}/login.jsp
${WELCOME URL}    http://${SERVER}/secure/Dashboard.jspa


*** Test Cases ***
Gherkin Valid Login
    [Tags]      WEB-1  WEB-3
    Given browser is opened to login page
    When user "admin" logs in with password "password123"
    Then welcome page should be open



*** Keywords ***
Browser is opened to login page
    Open browser to login page

Open Browser To Login Page
    Open Browser    ${LOGIN URL}    ${BROWSER}
    Maximize Browser Window
    Set Selenium Speed    ${DELAY}
    Login Page Should Be Open


User "${username}" logs in with password "${password}"
    Input username    ${username}
    Input password    ${password}
    Submit credentials

Input Username
    [Arguments]    ${username}
    Input Text    login-form-username    ${username}


Input Password
    [Arguments]    ${password}
    Input Text    login-form-password    ${password}

Welcome Page Should Be Open
    Location Should Be    ${WELCOME URL}
    Title Should Be    System Dashboard - Your Company JIRA
```

Robot-Framework example output XML


```
<?xml version="1.0" encoding="UTF-8"?>
<robot generated="20170220 14:18:54.562" generator="Robot 3.0.2 (Python 2.7.13 on win32)">
  <suite source="C:\Users\lmfv\Documents\Saco de Features\xray-1238\robot-example\robotframework-
webdemo\login_tests" id="s1" name="Login Tests">
    <suite source="C:\Users\lmfv\Documents\Saco de Features\xray-1238\robot-example\robotframework-
webdemo\login_tests\gherkin_login.robot" id="s1-s1" name="Gherkin Login">
      <test id="s1-s1-t1" name="Gherkin Valid Login">
        <kw name="Given browser is opened to login page">
          <kw name="Login Page Should Be Open" library="resource">
            <kw name="Title Should Be" library="Selenium2Library">
              <doc>Verifies that current page title equals `title`.</doc>
```

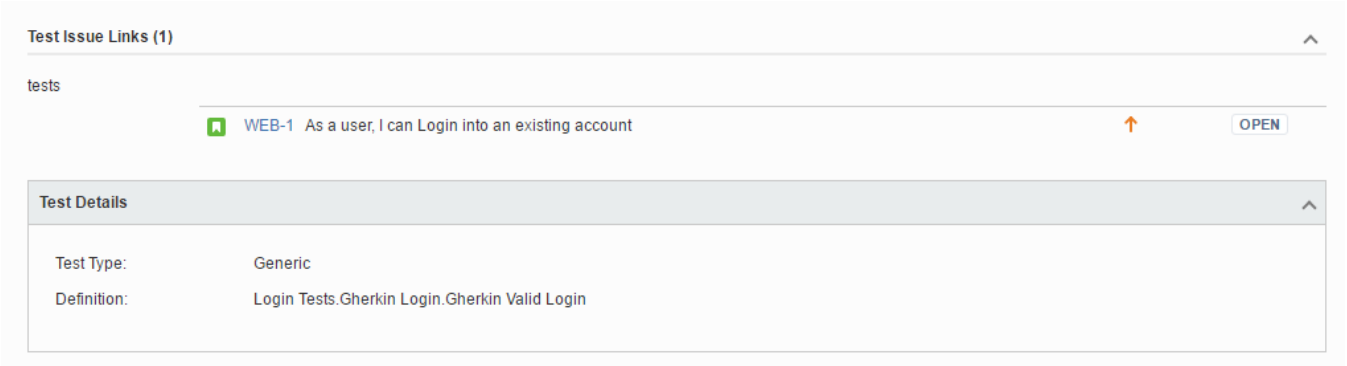
```
              <arguments>
                <arg>Log in - Your Company JIRA</arg>
              </arguments>
              <msg timestamp="20170220 14:19:07.693" level="INFO">Page title is 'Log in - Your Company JIRA'.<
/msg>
              <status status="PASS" endtime="20170220 14:19:07.693" starttime="20170220 14:19:07.158">
              </status>
            </kw>
            <status status="PASS" endtime="20170220 14:19:07.693" starttime="20170220 14:19:07.158">
            </status>
          </kw>
          <status status="PASS" endtime="20170220 14:19:07.693" starttime="20170220 14:18:55.937">
          </status>
        </kw>
        <kw name="When user &quot;admin&quot; logs in with password &quot;password123&quot;">
          <kw name="Input Username" library="resource">
            <arguments>
              <arg>${username}</arg>
            </arguments>
            <kw name="Input Text" library="Selenium2Library">
              <doc>Types the given `text` into text field identified by `locator`.</doc>
              <arguments>
                <arg>login-form-username</arg>
                <arg>${username}</arg>
              </arguments>
              <msg timestamp="20170220 14:19:07.696" level="INFO">Typing text 'admin' into text field 'login-
form-username'</msg>
              <status status="PASS" endtime="20170220 14:19:09.314" starttime="20170220 14:19:07.696">
              </status>
            </kw>
            <status status="PASS" endtime="20170220 14:19:09.314" starttime="20170220 14:19:07.695">
            </status>
          </kw>
          <kw name="Input Password" library="resource">
            <arguments>
              <arg>${password}</arg>
            </arguments>
            <kw name="Input Text" library="Selenium2Library">
              <doc>Types the given `text` into text field identified by `locator`.</doc>
              <arguments>
                <arg>login-form-password</arg>
                <arg>${password}</arg>
              </arguments>
              <msg timestamp="20170220 14:19:09.316" level="INFO">Typing text 'password123' into text field
'login-form-password'</msg>
              <status status="PASS" endtime="20170220 14:19:10.956" starttime="20170220 14:19:09.316">
              </status>
            </kw>
            <status status="PASS" endtime="20170220 14:19:10.956" starttime="20170220 14:19:09.315">
            </status>
          </kw>
          <kw name="Submit Credentials" library="resource">
            <kw name="Click Button" library="Selenium2Library">
              <doc>Clicks a button identified by `locator`.</doc>
              <arguments>
                <arg>login-form-submit</arg>
              </arguments>
              <msg timestamp="20170220 14:19:10.958" level="INFO">Clicking button 'login-form-submit'.</msg>
              <status status="PASS" endtime="20170220 14:19:17.476" starttime="20170220 14:19:10.958">
              </status>
            </kw>
            <status status="PASS" endtime="20170220 14:19:17.477" starttime="20170220 14:19:10.957">
            </status>
          </kw>
          <status status="PASS" endtime="20170220 14:19:17.478" starttime="20170220 14:19:07.695">
          </status>
        </kw>
        <kw name="Then welcome page should be open" library="resource">
          <kw name="Location Should Be" library="Selenium2Library">
            <doc>Verifies that current URL is exactly `url`.</doc>
            <arguments>
```

```
              <arg>${WELCOME URL}</arg>
            </arguments>
            <kw name="Capture Page Screenshot" library="Selenium2Library">
              <doc>Takes a screenshot of the current page and embeds it into the log.</doc>
              <msg timestamp="20170220 14:19:18.702" html="yes" level="INFO">&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;
&lt;td colspan="3"&gt;&lt;a href="selenium-screenshot-1.png"&gt;&lt;img src="selenium-screenshot-1.png" width="
800px"&gt;&lt;/a&gt;</msg>
              <status status="PASS" endtime="20170220 14:19:18.702" starttime="20170220 14:19:18.004">
              </status>
            </kw>
            <msg timestamp="20170220 14:19:18.705" level="FAIL">Location should have been 'http://localhost:8080
/secure/Dashboard.jspa' but was 'http://localhost:8080/login.jsp'</msg>
            <status status="FAIL" endtime="20170220 14:19:18.705" starttime="20170220 14:19:17.483">
            </status>
          </kw>
          <status status="FAIL" endtime="20170220 14:19:18.706" starttime="20170220 14:19:17.481">
          </status>
        </kw>
        <kw type="teardown" name="Close Browser" library="Selenium2Library">
          <doc>Closes the current browser.</doc>
          <status status="PASS" endtime="20170220 14:19:22.382" starttime="20170220 14:19:18.707">
          </status>
        </kw>
        <tags>
          <tag>WEB-1</tag>
          <tag>WEB-3</tag>
        </tags>
        <status status="FAIL" endtime="20170220 14:19:22.383" critical="yes" starttime="20170220 14:18:55.936"
>Location should have been 'http://localhost:8080/secure/Dashboard.jspa' but was 'http://localhost:8080/login.
jsp'</status>
      </test>
      <doc>A test suite with a single Gherkin style test.This test is functionally identical to the example
invalid_login.robot file.</doc>
      <status status="FAIL" endtime="20170220 14:19:22.397" starttime="20170220 14:18:54.670">
      </status>
    </suite>
    <status status="FAIL" endtime="20170220 14:22:12.549" starttime="20170220 14:18:54.567">
    </status>
  </suite>
</robot>
```


Entities

Each Robot test case is mapped to a Generic Test in Jira, having the summary with the name of the test case, and the **Generic Test Definition** field contains the concatenated names of the test suites, along with the name of the test case. Note that the Robot Framework considers the base folder of the project as the first test suite.

> **Please note**
>
> The way you run your tests also affects Robot's XML, so if you execute the file from somewhere else or you directly execute the file by passing it as an argument, the test suite's information will potentially be different.

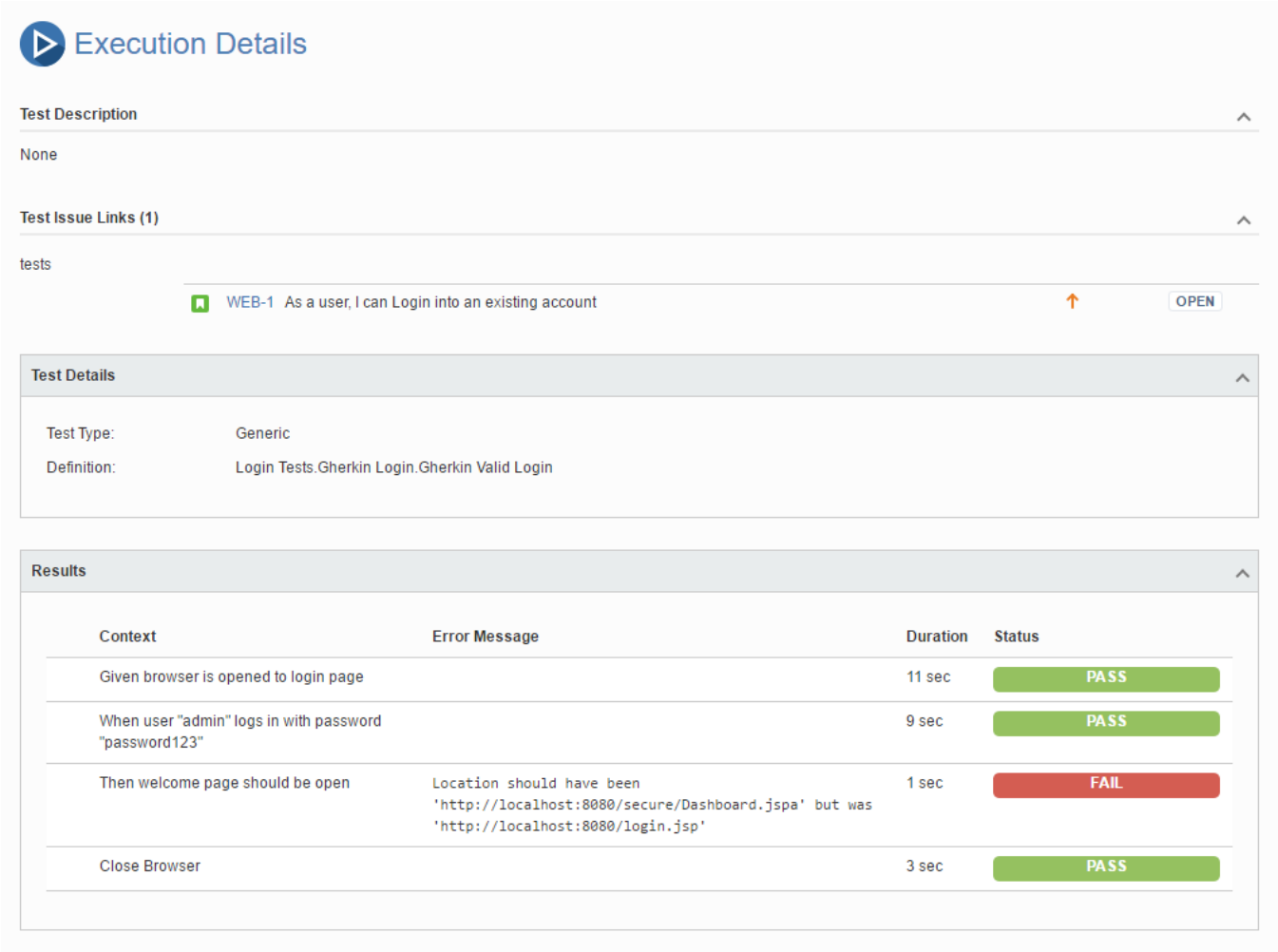


The name, error message, duration and status of Robot's keywords will be imported and stored in the test run results.

> **Note**
>
> The test case keywords and the test case setup/teardown keywords are individually imported; therefore, you can see their respective results. The test suite setup/teardown keywords are not imported.

Information about each Robot keyword (i.e., step), along with the corresponding status, are displayed in the Context section of the Execution Details of the Generic Test.

The example below shows the execution details page of a Test containing both test keywords as well as the final test teardown keyword.



Status

Robot Framework's test case status maps directly to Xray test run status FAIL and PASS.

Note: Test Cases with the status FAIL may have an error/failure message displayed in the Results section of the Test Run screen.

> **Working example**
>
> Please take a look at the tutorial Testing using Robot Framework integration in Python or Java for a more concrete end-to-end example.

References

- http://robotframework.org/
- https://github.com/robotframework/robotframework
- http://robotframework.org/robotframework/#user-guide