Robot Framework Sphinx Contrib Library Documentation

Release 0.4.3

asko.soukka@iki.fi

Sep 11, 2021

Contents

sphinxcontrib-robotframework is a Sphinx-extension, which executes embedded Robot Framework tests during `sphinx-build`.

sphinxcontrib-robotframework can be used in doctest way to validate examples shown in documentation or with Selenium and its Robot Framework integration, SeleniumLibrary, to generate scripted screenshots during the documentation compiliation time, for CI-generated up-to-date screenshots.

CHAPTER 1

Examples

1.1 Document with embedded tests

With the Robot Framework space separated format, a minimal test suite must contain the `*** Test Cases ***` header and at least one test case, like:

```
*** Test Cases ***

Foo is always foo
    Should be equal  foo  foo
```

But the `*** Test Cases ***`-header may be followed by as many tests as required, like:

```
*** Test Cases ***

Foo is still foo
    Should be equal  foo  foo

Foo is never bar
    Should not be equal  foo  bar
```

1.2 Document with a screenshot

The fun with **sphinxcontrib-robotframework** starts in using it together with SeleniumLibrary.

These packages together would allow you to navigate any website, take screenshots when required and finally embed those screenshot into this very Sphinx-documentation. All this with just `sphinx-build`:

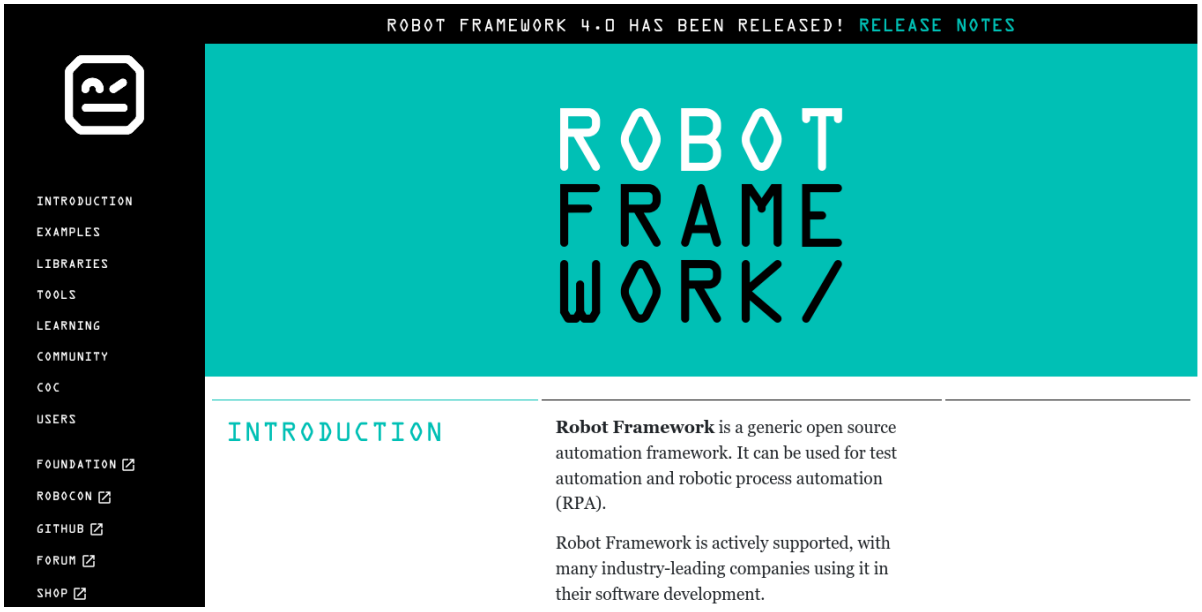
```
*** Settings ***

Library  SeleniumLibrary

Suite Teardown  Close all browsers

*** Variables ***
```

(continues on next page)



(continued from previous page)

```
${BROWSER}  headlessfirefox

*** Test Cases ***

Capture a screenshot of RobotFramework.org
    Open browser  http://robotframework.org/  browser=${BROWSER}
    Capture page screenshot  robotframework.png
```

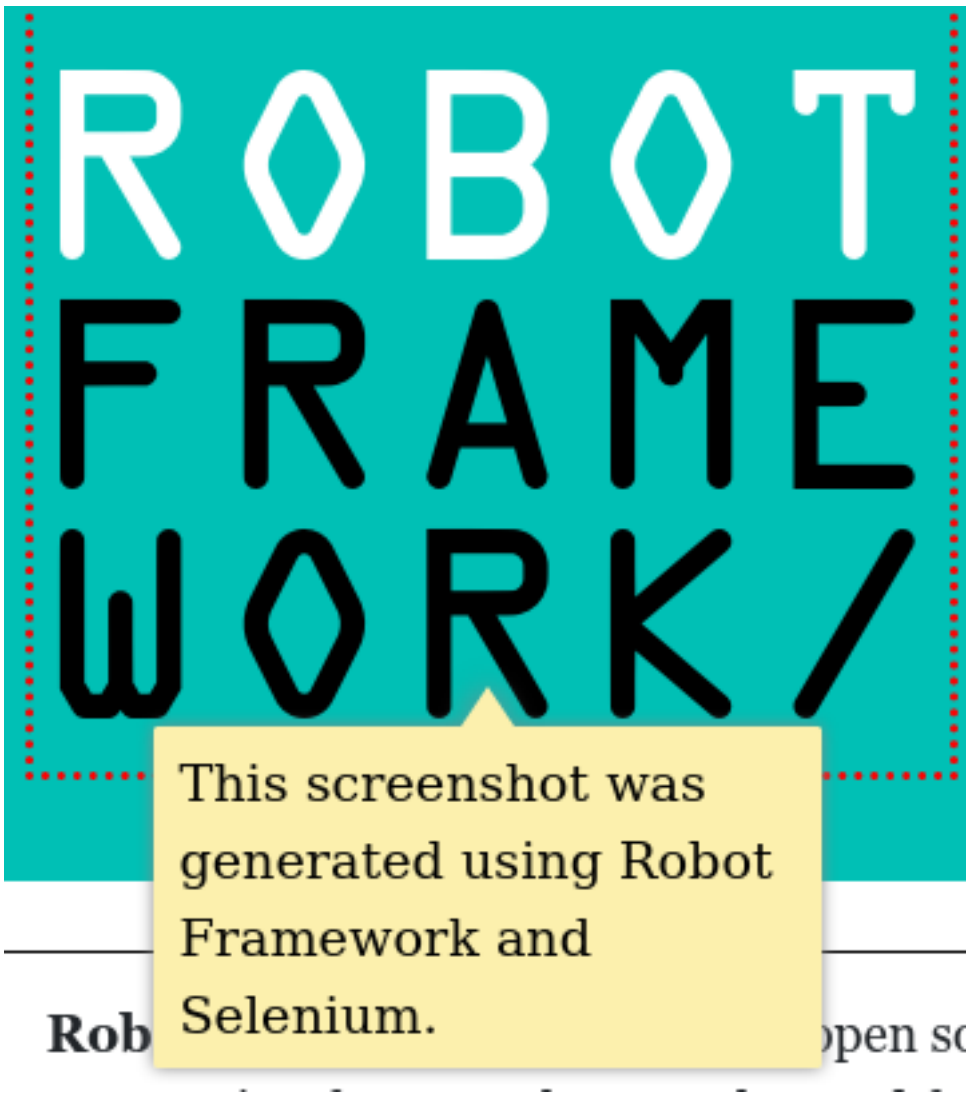
1.3 Document with an annotated screenshot

While Selenium has built-in support for capturing whole page screenshots, usually screenshots must be cropped and some times also annotated to make them useful in a documentation.

A Robot Framework library called SeleniumScreenshots provides a collection of re-usable keywords for cropping and annotating screenshots.

A cropped and annotated screenshot could look like this:

Note: The image cropping feature for **robotframework-seleniumscreenshots** requires PIL or Pillow.


ROBOT
FRAME
WORK/
This screenshot was generated using Robot Framework and Selenium.

CHAPTER 2

Getting started

1. Install **sphinxcontrib-robotframework** into your virtualenv or require it as a dependency of your Sphinx-project.
2. Enable the extension and execution of embedded Robot Framework tests by adding the following lines into your Sphinx-project's `conf.py`:

```
extensions = ['sphinxcontrib_robotframework']

# Enable Robot Framework tests during Sphinx compilation
sphinxcontrib_robotframework_enabled = True

# Hide Robot Framework syntax from the Sphinx output by default
# (preferred, when you use the extension for scripted screenshots)
sphinxcontrib_robotframework_quiet = True
```

3. Write your Robot Framework tests in space separated form as contents of Docutils' `code`-directives with `robotframework`-language:

```
.. code:: robotframework

   *** Settings ***

   ...

   *** Variables ***

   ...

   *** Test cases ***

   ...
```

   Each document may contain several `code`-directives, but their contents are concatenated into a single Robot Framework test suite before execution.

   The output of each `code`-directive can be omitted by setting a special `:class: hidden`-option. (This is not a standard Sphinx-behavior, but a hard coded feature in **sphinxcontrib-robotframework**.)

4. Compile your documentation and see your tests being run.

Note: If you choose to use Robot Framework variables in your test cases, you can override values for those variables in your Sphinx-configuration file (`conf.py`) with:

```
sphinxcontrib_robotframework_variables = {
    "VARIABLE": "value"
}
```

When Sphinx *nitpicky* mode is enabled, failing Robot Framework run will raise Sphinx Error and leave Robot Framework log files in place.