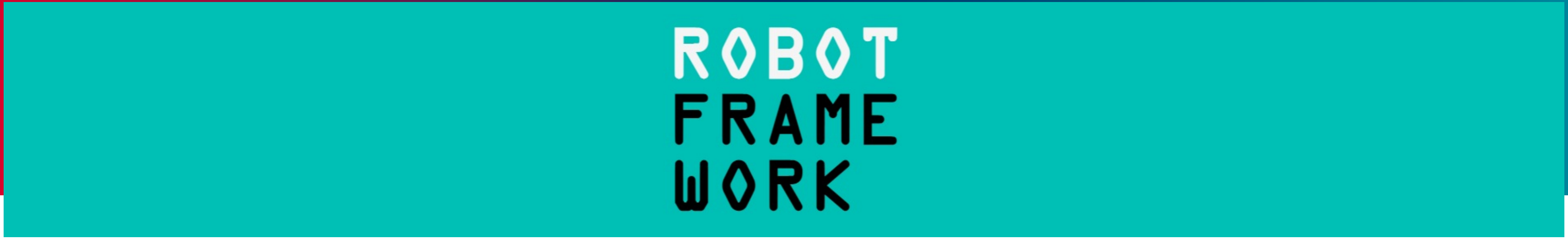
Workshop: Create a Robot Framework Keyword Library with Python

JS Jörg Schultze-Lutter | Last modified: 20.05.2022 | 6 minutes read



A brief introduction to using Python in combination with the Robot Framework and developing Robot keywords

Introduction

Robot Framework is a generic open source framework for automating tasks and can be used both to perform automated software testing with a focus on acceptance testing and for Robotic Process Automation (RPA). In addition to creating keyword libraries in Robot and using external Robot standard libraries, it is very easy to create your own Robot keyword libraries in languages such as Python to encapsulate sensitive code, for example. In general, anything that can be implemented as program code in Python can also be implemented as a Robot keyword library. The possibilities are therefore almost unlimited.

In this workshop we will create a Robot Framework library in Python and use it in a Robot test case. The accompanying sample code is **available in a Github repository**, so that you can immediately start creating our own libraries for the Robot Framework. Our Python robot library will be named `DemoRobotLibrary` and will support input and output parameters.

Prerequisites for this workshop

- Python 3.7 or later
- Robot Framework 4.0 or later (`pip install robotframework`)

Repository structure

Our **repository** consists of two directories:

- `tests` contains our final Robot Framework test suite that uses our Python Robot Keyword library.
- `DemoRobotLibrary` is the directory for our Python robot library. It is recommended to always choose the directory name analogous to the library name.

Functional scope of our Python Robot keyword library

Our library supports the following feature set:

Input parameters:

- User's name
- optional: language (ISO639 a2). Default is English; supported are English and German in total.

Output parameter:

- Greeting (string). A return string is created based on the local time, name and language.

Create our Robot library with Python

First, we create the basic framework of our library and create our two Python files.

__init__.py (for the creation of a later Python package):

```
from .DemoRobotLibrary import DemoRobotLibrary
```

DemoRobotLibrary.py (this is the library itself):

Here we first start with our standard includes. Here I use the standard Python logger for local tests and for the integration into the Robot Framework the necessary Robot imports and datetime for the determination of the current time.

```
from robot.api.deco import library, keyword, not_keyword
from datetime import datetime
import logging

logging.basicConfig(
    level=logging.DEBUG, format="%(asctime)s %(module)s -%
(levelname)s- %(message)s"
)
logger = logging.getLogger(__name__)
```

After we have successfully mastered this step, it is now time to create the actual library. This is created as a Python class. We thus create our class and create or initialize our class variables. As we want to turn this code into a Robot library, we declare the necessary code via a decorator which is part of the `robot.api.deco` library. For our keyword library, we specify a global scope and deactivate the automatic creation of robot keywords from our Python library, so that only those Python functions with a `@keyword` decorator are exposed as Robot keywords. Use this as a 'best practice' approach even when `auto_keywords=False` is used by your library.

```
@library(scope="GLOBAL", auto_keywords=False)
class DemoRobotLibrary:
    # Default parameter settings
    DEFAULT_NAME = None
    DEFAULT_LANGUAGE = "en"

    # Class-internal parameters
    __demo_name = None
    __demo_language = None
```

We then initialize our class.

```
def __init__(
    self,
    demo_name: str = DEFAULT_NAME,
    demo_language: str = DEFAULT_LANGUAGE,
):
    self.__demo_name = demo_name
    self.__demo_language = demo_language
```

Now we create the Python specific "getter"/"setter" methods. For the "setter" methods we check if a value has been passed. Additionally we check for a valid language code.

```
# Python "Getter"
@property
def demo_name(self):
    return self.__demo_name

@property
def demo_language(self):
    return self.__demo_language

# Python "Setter"
@demo_name.setter
def demo_name(self, demo_name: str):
    if not demo_name:
        raise ValueError("No 'name' value has been specified")
    self.__demo_name = demo_name

@demo_language.setter
def demo_language(self, demo_language: str):
    if not demo_language:
        raise ValueError("No 'language' value has been specified")
    demo_language = demo_language.lower()
    if demo_language not in ['de','en']:
        raise ValueError("Invalid value for 'language' was specified")
    self.__demo_language = demo_language
```

With these steps, we have completed the preliminary work for creating our library. Now we move on to the creation of our Robot keywords. Again, we start with Robot-specific "getter" and "setter" keywords, each of which we define using a decorator. Validity checks for the individual parameters are automatically delegated to our class - so we don't have to worry about them. The parameter names used for the setters correspond to the parameter names used later in Robot.

```
    # Robot "Getter" Keywords
    @keyword("Get My Name")
    def get_demo_name(self):
        return self.demo_name

    @keyword("Get My Language")
    def get_demo_language(self):
        return self.demo_language

    # Robot "Setter" Keywords
    @keyword("Set My Name")
    def set_demo_name(self, name: str = None):
        logger.debug(msg="Setting Demo 'name'")
        self.demo_name = name

    @keyword("Set My Language")
    def set_demo_language(self, language: str = None):
        logger.debug(msg="Setting Demo 'language'")
        self.demo_language = language
```

Then we create a separate keyword, which determines a greeting depending on the time of day based on the name and the language setting. The actual determination is done by a subroutine, which we explicitly do not publish as a robot keyword in our library (`@not_keyword` decorator).

```
    @not_keyword
    def get_daytime_salutation(self,language: str):
        retval = None

        a = datetime.now().hour
        if a < 6 or a > 23:
            retval = "Gute Nacht" if language == "de" else "Good Night"
        elif a >= 6 and a <= 11:
            retval = "Guten Morgen" if language == "de" else "Good Day"
        elif a >=12 and a <= 17:
            retval = "Guten Nachmittag" if language == "de" else "Good
    Afternoon"
        elif a >=17 and a <= 22:
            retval = "Guten Abend" if language == "de" else "Good Evening"
        else:
            retval = "Guten Tag" if language == "de" else "Good Day"

        return retval

    @keyword("Get My Salutation")
    def get_the_salutation_text(self, name: str, language: str=None):
    logger.debug(msg="Get the salutation text")

    # Get our language. The language code can be taken from either this
    keyword or from the class.
    # At least one language code needs to be set
    if not self.demo_language and not language:
        raise ValueError ("No language specified")
    lang = language if language else self.demo_language

    localized_salutation = get_daytime_salutation(language=lang)
    return_value = f"{localized_salutation} {name}!"

    return return_value
```

Our Robot Framework library is ready and we can now include it in a Robot Framework test case.

Create our Robot Test Case

```
*** Settings ***
Library   DemoRobotLibrary

*** Test Cases ***
My Successful Test Case

    # Get the salutation string with the class defaults
    # Returns an English salutation string
    ${SALUTATION}=  Get My Salutation   Peter
    Log To Console  ${SALUTATION}

    # Get the Salutation and use named parameters
    ${SALUTATION}=  Get My Salutation   name=John
    Log To Console  ${SALUTATION}

    # Get the Salutation, use named parameters and override the
language defaults
    ${SALUTATION}=  Get My Salutation   name=Franz  language=de
    Log To Console  ${SALUTATION}

    # Set the language and then get the salutation
    Set My Language   language=de
```

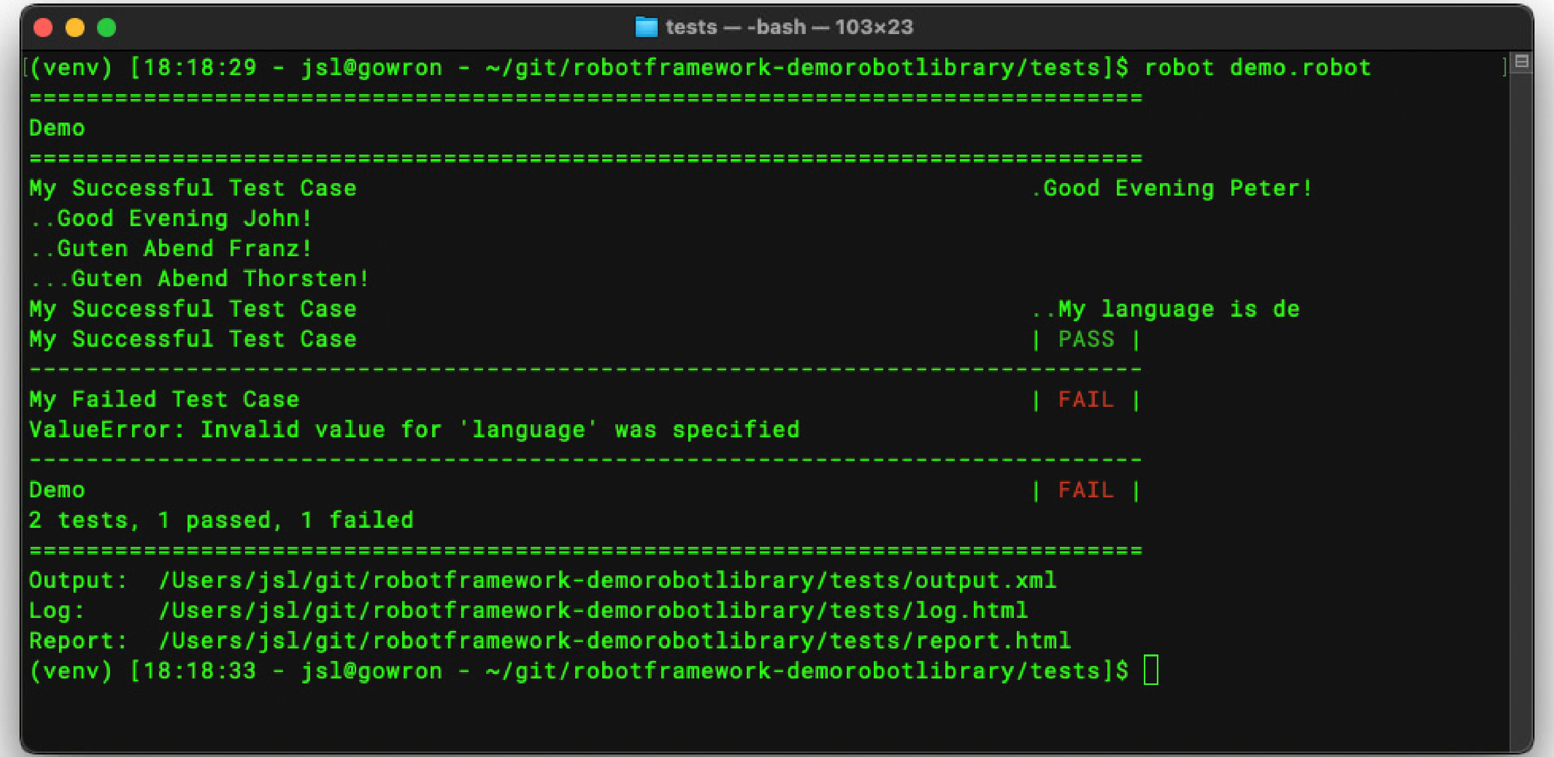
```
        ${SALUTATION}=  Get My Salutation   name=Thorsten
        Log To Console  ${SALUTATION}

        # Get the language
        ${LANGUAGE}=  Get My Language
        Log To Console  My language is ${LANGUAGE}

    My Failed Test Case
        # Set an invalid language which will fail this test
        Set My Language   language=fr
```

Run our Robot test case

To be able to run our robot test without publishing our Robot library to the PyPi index, we need to point the `PYTHONPATH` variable to the directory where our Python library is located:

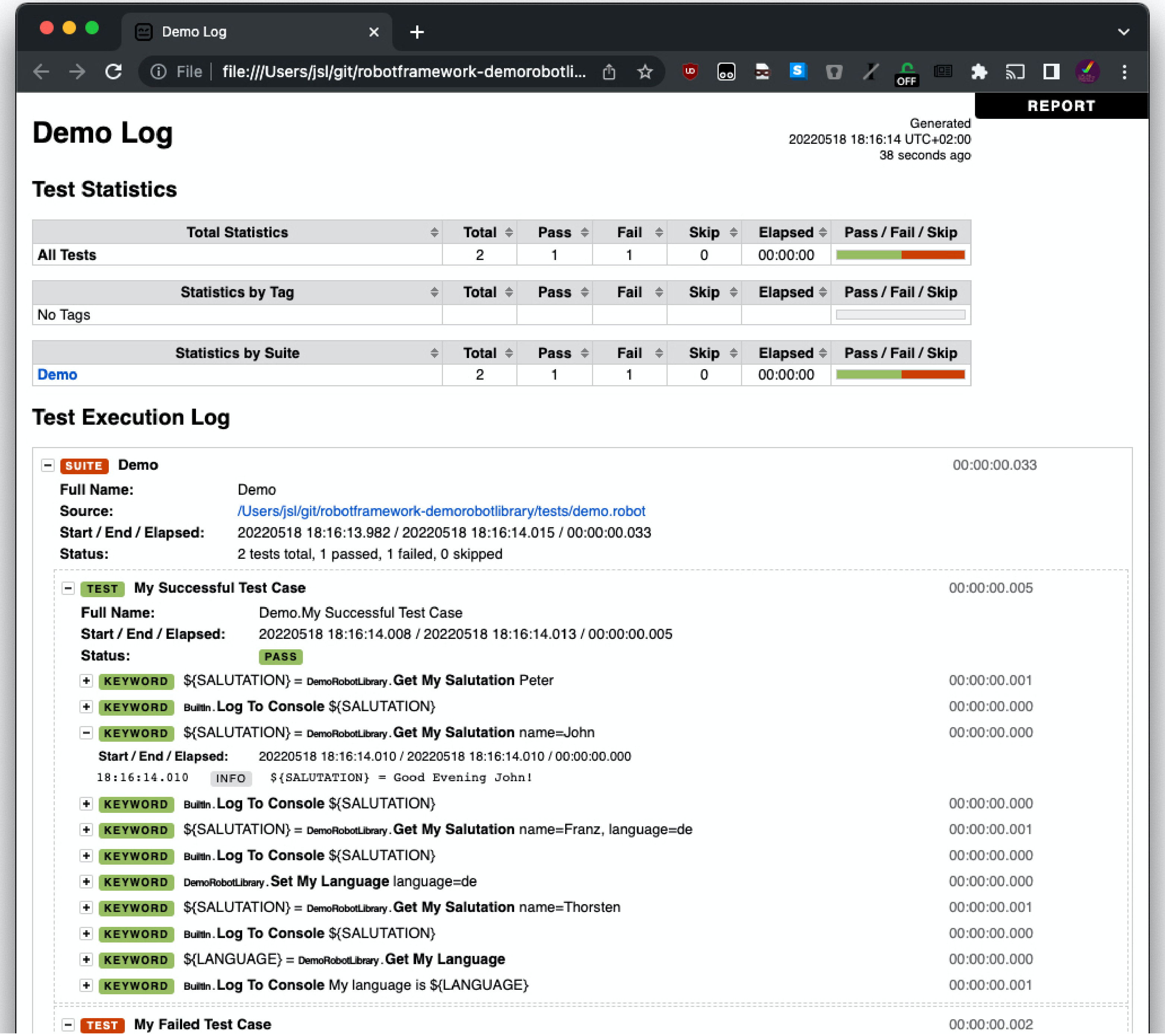
Linux: `export PYTHONPATH=/Users/jsl/robotframework-demorobotlibrary/DemoRobotLibrary`
Windows: `set PYTHONPATH=C:\Users\jsl\robotframework-demorobotlibrary\DemoRobotLibrary`

Now let's run our Robot test suite with `robot demo.robot`.



```
(venv) [18:18:29 - jsl@gowron - ~/git/robotframework-demorobotlibrary/tests]$ robot demo.robot
==============================================================================
Demo
==============================================================================
My Successful Test Case                                               .Good Evening Peter!
..Good Evening John!
..Guten Abend Franz!
...Guten Abend Thorsten!
My Successful Test Case                                               ..My language is de
My Successful Test Case                                               | PASS |
------------------------------------------------------------------------------
My Failed Test Case                                                   | FAIL |
ValueError: Invalid value for 'language' was specified
------------------------------------------------------------------------------
Demo                                                                  | FAIL |
2 tests, 1 passed, 1 failed
==============================================================================
Output:  /Users/jsl/git/robotframework-demorobotlibrary/tests/output.xml
Log:     /Users/jsl/git/robotframework-demorobotlibrary/tests/log.html
Report:  /Users/jsl/git/robotframework-demorobotlibrary/tests/report.html
(venv) [18:18:33 - jsl@gowron - ~/git/robotframework-demorobotlibrary/tests]$
```

We can see that as expected the keywords of our new Python Robot library can be used. The Python code used is not included in the logfile of our Robot test suite. In addition, our second test fails when we try to pass an unsupported language code to our library. Our keyword library works as expected.



Demo Log

Generated
20220518 18:16:14 UTC+02:00
38 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	2	1	1	0	00:00:00	🟩🟥

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

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Demo	2	1	1	0	00:00:00	🟩🟥

Test Execution Log

SUITE **Demo** 00:00:00.033

Full Name: Demo
Source: /Users/jsl/git/robotframework-demorobotlibrary/tests/demo.robot
Start / End / Elapsed: 20220518 18:16:13.982 / 20220518 18:16:14.015 / 00:00:00.033
Status: 2 tests total, 1 passed, 1 failed, 0 skipped

TEST **My Successful Test Case** 00:00:00.005

Full Name: Demo.My Successful Test Case
Start / End / Elapsed: 20220518 18:16:14.008 / 20220518 18:16:14.013 / 00:00:00.005
Status: PASS

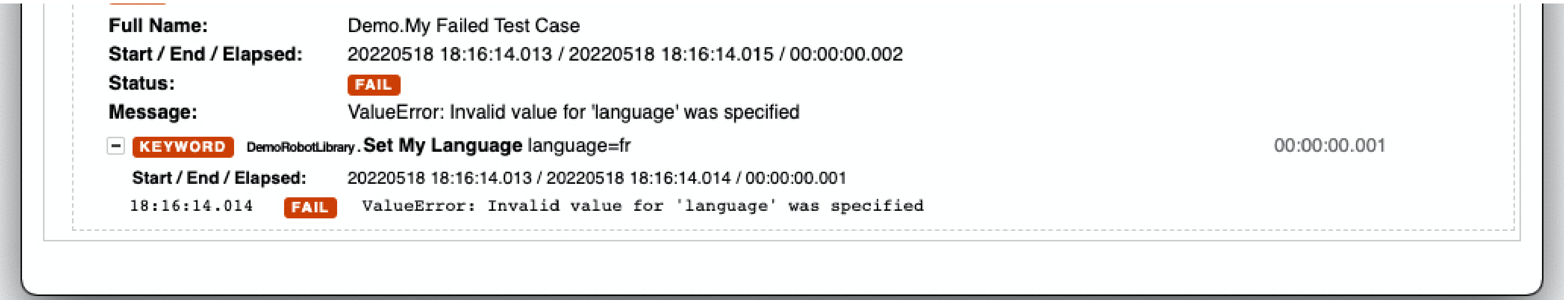
KEYWORD ${SALUTATION} = DemoRobotLibrary.**Get My Salutation** Peter 00:00:00.001
KEYWORD BuiltIn.**Log To Console** ${SALUTATION} 00:00:00.000
KEYWORD ${SALUTATION} = DemoRobotLibrary.**Get My Salutation** name=John 00:00:00.000

Start / End / Elapsed: 20220518 18:16:14.010 / 20220518 18:16:14.010 / 00:00:00.000

```
18:16:14.010  INFO  ${SALUTATION} = Good Evening John!
```

KEYWORD BuiltIn.**Log To Console** ${SALUTATION} 00:00:00.000
KEYWORD ${SALUTATION} = DemoRobotLibrary.**Get My Salutation** name=Franz, language=de 00:00:00.001
KEYWORD BuiltIn.**Log To Console** ${SALUTATION} 00:00:00.000
KEYWORD DemoRobotLibrary.**Set My Language** language=de 00:00:00.000
KEYWORD ${SALUTATION} = DemoRobotLibrary.**Get My Salutation** name=Thorsten 00:00:00.001
KEYWORD BuiltIn.**Log To Console** ${SALUTATION} 00:00:00.000
KEYWORD ${LANGUAGE} = DemoRobotLibrary.**Get My Language** 00:00:00.000
KEYWORD BuiltIn.**Log To Console** My language is ${LANGUAGE} 00:00:00.001

TEST **My Failed Test Case** 00:00:00.002


```
Full Name:              Demo.My Failed Test Case
Start / End / Elapsed:  20220518 18:16:14.013 / 20220518 18:16:14.015 / 00:00:00.002
Status:                 FAIL
Message:                ValueError: Invalid value for 'language' was specified
− KEYWORD  DemoRobotLibrary.Set My Language language=fr                              00:00:00.001
   Start / End / Elapsed:  20220518 18:16:14.013 / 20220518 18:16:14.014 / 00:00:00.001
   18:16:14.014   FAIL   ValueError: Invalid value for 'language' was specified
```

Thus, our library could theoretically be published on platforms like PyPi. This process was already described in one of my previous BCP articles.

Good luck with developing your own keyword libraries for the Robot Framework!

The opinions and information stated in this article are personal to the individual author and do not necessarily represent Bertelsmann.

About the Author

JS

Jörg Schultze-Lutter

Arvato