This blog is for me a platform to share my knowledge in the area of software development.

DEVOPS

Improve Your Robot Framework Tests With Robocop

Date: March 31, 2021 Author: mydeveloperplanet 2 Comments

Testing software will tell you something about the quality of your software. But how do you ensure that your tests have a high quality? A static analysis tool can help you in order to improve the quality. Robocop is a such a static analysis tool for Robot Framework tests. In this blog, you will learn how to use and configure Robocop for your purposes.

1. Introduction

Static code analyzers are commonly used during software development. SonarQube is a popular tool used for this purposes. The intention is to scan the code and verify whether it complies to a certain set of rules. The rules to check against should be agreed upon within your company and serve as best practices. The purpose of the scan is to learn, to standardize, to improve readability, etc.

Robocop is such a static analysis tool for Robot Framework tests. In case you need a basic overview of the capabilities of Robot Framework, you can read some previous blogs as an introduction:

- Automated Acceptance Testing With Robot Framework
- How to Write Data Driven Tests With Robot Framework
- Create Custom Robot Framework Libraries

The Robocop documentation gives you a good overview of all capabilities. In the sections below, you will learn how to use the tool from a practical point of view. This way, it should be quite easy for you to apply it to your own Robot Framework test cases. As an example, you can use the GitHub repository from the previous blogs which contain a set of test cases and resource files. The `master` branch contains non-analyzed tests, the `feature/robocop` branch contains fixes for the Robocop warnings.

First of all, you will need to install Robocop. At the time of writing, Robocop v1.5 is the latest release.

```
$ pip install robotframework-robocop
```

2. Run a Scan

Let’s see what happens when you run a Robocop scan from within the root of the repository. Robocop will search for test cases and resource files in the current directory and in the subdirectories and will run with default settings. The output shows you all the files where violations against the rules have been found. The output format gives you the following information:

- The file name where the violation has been found;
- The line number where the violation occured;
- The column where the violation has been detected;
- The severity of the violation (E for error, W for warning, I for information);
- The ID of the rule;
- The description of the rule.

```
$ robocop
<path>/test/employee.robot:1:0 [W] 0704 Ignored data found in file
<path>/test/employee.robot:4:0 [W] 1003 Invalid number of empty line
<path>/test/employee.robot:7:0 [W] 1003 Invalid number of empty line
<path>/test/employee.robot:9:13 [W] 1007 Line is over-indented
<path>/test/employee.robot:11:15 [W] 0906 Redundant equal sign in va
...
```

In order to limit the number of violations, it is possible to limit the scan to one specific file. This way, only the results for this particular file will be shown.

```
$ robocop test/employee.robot
```

Robocop also provides the option to generate a report of the issues found. It will give you a summary which provides a nice overview.

```
$ robocop --report all test/employee.robot
<path>/test/employee.robot:1:0 [W] 0704 Ignored data found in file
...
Processed 1 file(s) from which 1 file(s) contained issues

Issues by IDs:
W1007 (uneven-indent)                   : 6
W0906 (redundant-equal-sign)            : 5
W1003 (empty-lines-between-sections)    : 3
W0302 (not-capitalized-keyword-name)    : 3
W0704 (ignored-data)                    : 1
W0508 (line-too-long)                   : 1
W1002 (missing-trailing-blank-line)     : 1

Found 20 issue(s): 20 WARNING(s).

Scan took 0.006s
```

3. More About the Rules

You now have an overview of the issues being found, but what are the rules being checked against? This is well documented in the official documentation. Another way of showing this list, is by means of the following command:

```
$ robocop --list
Rule - 0201 [W]: missing-doc-keyword: Missing documentation in keywo
Rule - 0202 [W]: missing-doc-testcase: Missing documentation in test
Rule - 0203 [W]: missing-doc-suite: Missing documentation in suite (
...
```

It might be the case that you do not agree with the severity of a certain rule. Let us assume that you do not agree with the severity of the `line-too-long` rule. This rule has a severity **Warning**, but you want to change it to severity **Information**. With Robocop it is possible to change this behaviour by means of the configuration argument. You just need to use the **ID** or **message name** of the rule, followed by argument `severity`, followed by the new value.

```
$ robocop -c line-too-long:severity:i --report all test/employee.rob
...
<path>/test/employee.robot:34:0 [I] 0508 Line is too long (138/120)
...
Issues by IDs:
...
I0508 (line-too-long)                : 1
...
```

In the output you can now see that the issue is still reported, but now with severity **Information**.

Not only the severity can be changed. A lot of rules have configurable parameters which can be changed according to your preferences. In order to know what these parameters are, you can take a look at the documentation. But it is also possible to show this information by means of the `list-configurables` command followed by the **ID** or **message name** of the rule.

```
$ robocop --list-configurables line-too-long
Rule - 0508 [W]: line-too-long: Line is too long (%d/%d) (enabled)
    Available configurable(s) for this rule:
        severity
        line_length
```

Besides the severity, it is also possible to change the line length of the `line-too-long` rule. Just like the severity, you change this parameter to e.g. 160. Running the scan with this setting, no issue is raised now because the line length is within the limits you specified.

```
$ robocop -c line-too-long:line_length:160 --report all test/employee
...
Found 19 issue(s): 19 WARNING(s).
```

As you can see, one issue less is shown in the total number of issues and in your output, no `line-too-long` warning is shown anymore.

It might be that you do not agree with a certain rule at all. In this case, you can disable a rule from being checked. You can do so by means of the `exclude` argument. Again, this will result in 19 warnings for the `employee.robot` test case.

```
$ robocop --exclude 0508 --report all test/employee.robot
...
Found 19 issue(s): 19 WARNING(s).
```

Listing the rules, you notice that rule `0508` is now indicated as being disabled.

```
$ robocop --exclude 0508 --list
...
Rule - 0508 [W]: line-too-long: Line is too long (%d/%d) (disabled)
...
```

In another case, you might want the rule to be enabled by default, but you want to disable an issue. You have noticed the issue and you are ok with it and do not want to be triggered about this part again in the future. You can disable a certain part of your test case for a specific rule. It is not advised to use this regularly, it will not improve the readability of your test case. It is better to fix the issue or to disable the rule or to reconfigure the rule for your needs.

```
# robocop: disable=line-too-long
| | ${rc}                              | ${output} =      | Run and Return R
# robocop: enable
```

4. Use a Configuration File

Instead of tweaking the configuration of rules in the command line, you can add the configuration in a configuration file. This way you can ensure that everyone is using the same configuration, certainly when you put it under version control. The contents of this file is for example:

```
--exclude 0508
--report all
```

Now we run robocop with the `argumentfile` argument followed by the file containing the arguments, `robocop_args.txt` in this case.

```
$ robocop --argumentfile robocop_args.txt  test/employee.robot
```

Robocop also offers the possibility to read the configuration file by default. This is restricted under the following conditions:

- The argument file must be named `.robocop`
- Robocop must be run without **any** other arguments. Note that it may not contain any other argument, also no file to analyze for example. It only works when only the `robocop` command is being used.

The `.robocop` argument file has the following content when you follow the above example.

```
--exclude 0508
--report all
test/employee.robot
```

Now it is sufficient to use the `robocop` command. In the output you can see that the default configuration file has been loaded.

```
$ robocop
Loaded default configuration file from '<path>/.robocop'
Processed 1 file(s) from which 1 file(s) contained issues

Issues by IDs:
No issues found

Found 0 issues

Scan took 0.005s
```

5. Fix Issues

In this section, some experiences are shared when fixing the issues in the test cases in the repository.

In the `employee.robot` file, all lines containing a `Documentation` tag raise the over-indented warning.

```
employee.robot:11:4 [W] 1007 Line is over-indented
```

Tried a lot of things in order to fix this, but this was not successful. Raised an issue for this and received a very quick response from the Robocop maintainers. It seems that the rule is ok, but when fixing it, another warning is shown (under-indented warning). This last one is not correct, but is already fixed in a Pull Request and will be available in a next release.

In the `data_driver.robot` file, a warning is raised about a missing documentation in the test case. In this case, it is ok to disable the rule, because this is the way the Data Driver library works. You cannot fix this.

```
*** Test Cases ***
# robocop: disable=missing-doc-testcase
| Add Employee ${first_name} ${last_name}
# robocop: enable
```

Before starting using Robocop, take a good look at the rules you want to enable or disable. Not every rule will be suitable for you and it is better to only enable those rules you really want to check upon. When too many rules are enabled which do not increase your overall quality, but are reported every time, people will tend to not using it anymore.

6. Conclusion

Robocop is a great initiative which can be of good help when you want to increase the quality of your Robot Framework scripts. Beware of which rules you enable, you do not want to be overflooded by warnings you do not care about.

ROBOCOP ROBOT FRAMEWORK TESTING TUTORIAL

Published by mydeveloperplanet

View all posts by mydeveloperplanet

2 thoughts on “Improve Your Robot Framework Tests With Robocop”

Add Comment

1. **Uli Seidel says:**
   August 24, 2021 at 1:33 pm
   Thanks for your blog.
   I took your example and tested with
   robocop –exclude missing-doc-testcase test.robot
   and got the message
   ### DEPRECATION WARNING: The name of the rule ‘missing-doc-testcase’ is renamed to ‘missing-doc-test-case’ starting from Robocop 1.8.0. Update your configuration if you’re using old name. ###
   Would you check whether the occurences of missing-doc-testcase should be replaced by missing-doc-test-case, please?
   I made some tests with VSCode and the .robocop configuration file and wondered why the excluded rule had no effect on the linted coding (until I changed missing-doc-testcase to missing-doc-test-case).
   Kind regards
   Uli

   Reply

   1. **mydeveloperplanet says:**
      August 25, 2021 at 5:59 pm
      Thank you for your comment. You are correct about the changed rule. We have noticed it ourselves at work after upgrading Robocop. It seems that the authors now and then change rules, but you will receive a clear deprecation warning of it. The current set of rules can be found at https://robocop.readthedocs.io/en/latest/rules.html and here you notice that version 1.9.0 contains the rule missing-doc-test-case. The release notes can be found at https://github.com/MarketSquare/robotframework-robocop/releases where you can see that this was one of the changes in v1.8.0.

      Reply

This site uses Akismet to reduce spam. Learn how your comment data is processed.