`

Contents

`

1. Basic information

Polarion Report Maker is application prepared for integration automatic test execution with test management in Polarion. The library is implemented in Python programming language and used in Pytest framework allows creating test reports directly linked with test cases from Polarion and creating test runs automatically after test session executed. Minimal Python for using the tool is 3.9.

Polarion Report Maker GIT repository:

https://github.com/DiehlControlsSoftware/internal-pypkg-dc-polarion_report_maker

DevPi server with Polarion Report Maker library:

https://devpi.diehlako.local/controls/prod

2. Released versions

Library name	Version	Features and changes
dc-polarion_report_maker	*1.0.0*	• *linking test cases between Polarion and created test pytest-html reports using API or exported files*
dc-polarion_report_maker	*1.1.0*	• *linking test cases between Polarion and created test pytest-html reports using API or exported files* • *automatic creation of test runs in Polarion after test session executed*
dc-polarion_report_maker	1.2.0	• *linking test cases between Polarion and created test pytest-html reports using API or exported files* • *automatic creation of test runs in Polarion after test session executed* • *library improvements*

3. Project library installation

3.1. How to install library in project using command line

The method of installing the library in a project can be found directly in ***README.md*** in *"Advices how to use/ install the Releases"* section in Github library repository:

https://github.com/sucheckij/report_maker

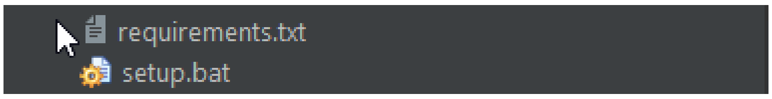
Note: During test project creation that method is more complicated than next one, so there is preference to skip that one and go to point 3.2.

That method can be more useful during just created Python project and can be more useful during adding Polarion Report Maker to already existing project.

`

3.2. How to install library using *setup.bat* and *requirements.txt*

In main folder of test project create files:

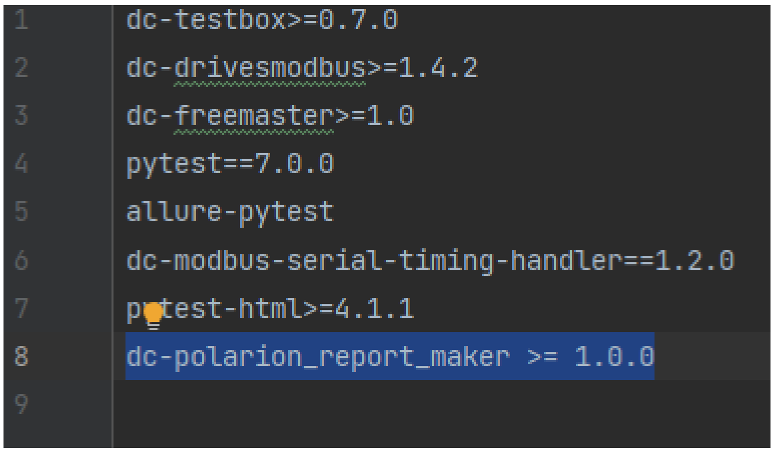


In ***setup.bat*** create a command sequence:

Copy/Paste script:

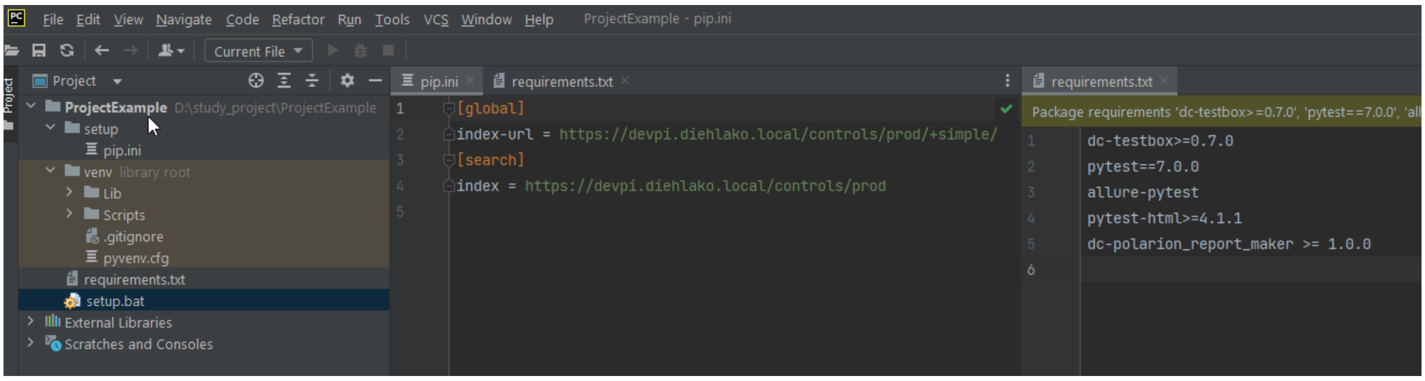
```
call venv\Scripts\deactivate
rmdir /S /Q venv
python -m venv venv
call venv\Scripts\activate
python -m pip install --upgrade pip
python -m pip install -r requirements.txt
```

In ***requirements.txt*** add *dc-polarion_report_maker* library with correct version:



```
dc-testbox>=0.7.0
dc-drivesmodbus>=1.4.2
dc-freemaster>=1.0
pytest==7.0.0
allure-pytest
dc-modbus-serial-timing-handler==1.2.0
pytest-html>=4.1.1
dc-polarion_report_maker >= 1.0.0
```

Initial project structure should be prepared as below in the picture:



```
[global]
index-url = https://devpi.diehlako.local/controls/prod/+simple/
[search]
index = https://devpi.diehlako.local/controls/prod
```

Package requirements 'dc-testbox>=0.7.0', 'pytest==7.0.0', 'all

```
dc-testbox>=0.7.0
pytest==7.0.0
allure-pytest
pytest-html>=4.1.1
dc-polarion_report_maker >= 1.0.0
```

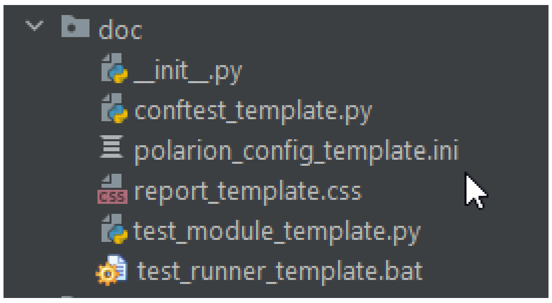
Run ***setup.bat***. Just after commands execution venv environment and project libraries are installed succesfully. Polarion Report Maker appliance is now part of the project and it is ready to use.

Note: Remember that .bat files works properly from CMD (but not from PowerShell)

`

4. Library elements of dc-polarion_report_maker

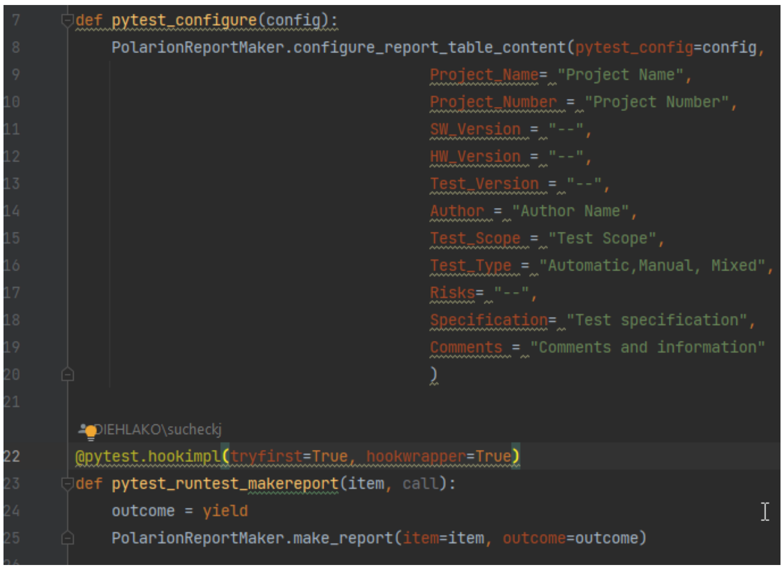
Just Polarion Report Maker installation all important items to include that tool into test project can be found in location: ***venv -> Lib -> polarion_report_maker_doc.***



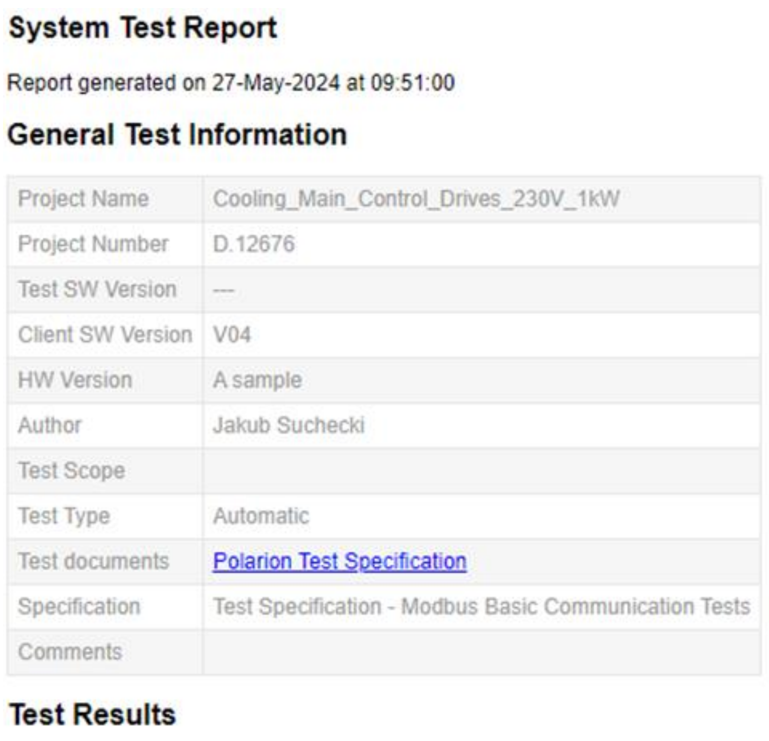
```
doc
    __init__.py
    conftest_template.py
    polarion_config_template.ini
    report_template.css
    test_module_template.py
    test_runner_template.bat
```

The file structure includes:

- **Conftest_template.py –** (conftest.py) configuration file which should be included each test module folder or globally for all tests. That file handle configuration and test report creation during test session executions. In that file you can configure or add all details which should be included in final test report information. That functions should not be modified (can be only extended by new functionalities



```
def pytest_configure(config):
    PolarionReportMaker.configure_report_table_content(pytest_config=config,
                                                       Project_Name= "Project Name",
                                                       Project_Number = "Project Number",
                                                       SW_Version = "--",
                                                       HW_Version = "--",
                                                       Test_Version = "--",
                                                       Author = "Author Name",
                                                       Test_Scope = "Test Scope",
                                                       Test_Type = "Automatic,Manual, Mixed",
                                                       Risks= "--",
                                                       Specification= "Test specification",
                                                       Comments = "Comments and information"
                                                       )

DIEHLAKO\sucheckj
@pytest.hookimpl(tryfirst=True, hookwrapper=True)
def pytest_runtest_makereport(item, call):
    outcome = yield
    PolarionReportMaker.make_report(item=item, outcome=outcome)
```



System Test Report

Report generated on 27-May-2024 at 09:51:00

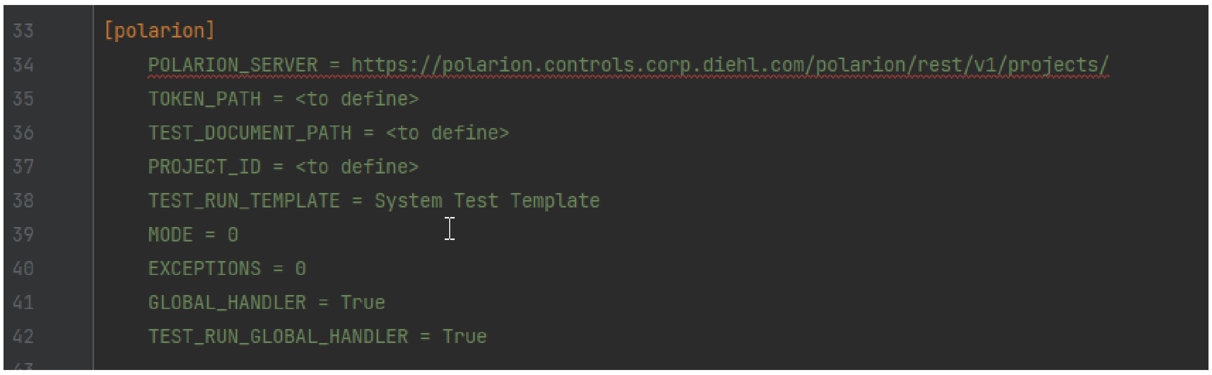
General Test Information

Project Name	Cooling_Main_Control_Drives_230V_1kW
Project Number	D.12676
Test SW Version	---
Client SW Version	V04
HW Version	A sample
Author	Jakub Suchecki
Test Scope	
Test Type	Automatic
Test documents	Polarion Test Specification
Specification	Test Specification - Modbus Basic Communication Tests
Comments	

Test Results

`

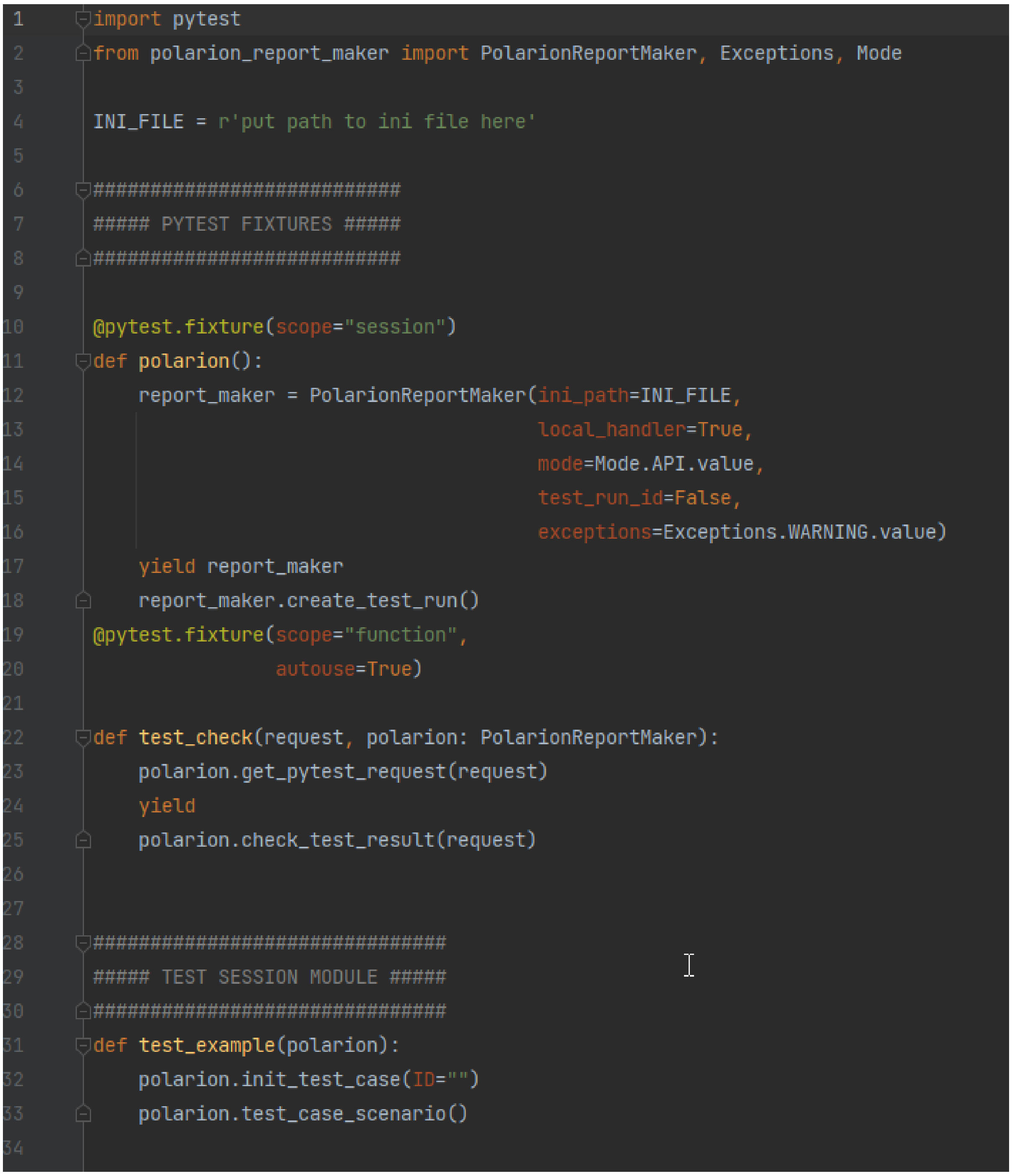
- ***polarion_config_template.ini* –** (polarion_config.ini) configuration file of Polarion Report Maker. All details how to configure parameters you can find directly in that file in comments sections (General info, settings, parameters usage)



```
[polarion]
    POLARION_SERVER = https://polarion.controls.corp.diehl.com/polarion/rest/v1/projects/
    TOKEN_PATH = <to define>
    TEST_DOCUMENT_PATH = <to define>
    PROJECT_ID = <to define>
    TEST_RUN_TEMPLATE = System Test Template
    MODE = 0
    EXCEPTIONS = 0
    GLOBAL_HANDLER = True
    TEST_RUN_GLOBAL_HANDLER = True
```

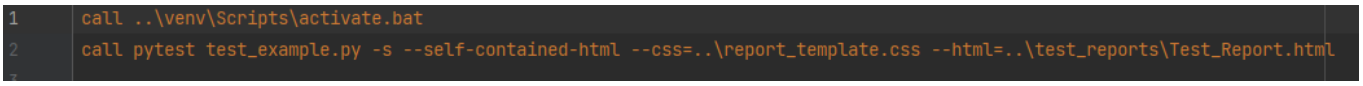
`

- ***test_module_template.py* –** test module of Pytest framework to use Polarion Report Maker as a part of tests in project. This is a template and can be used as the initial file to create automatic tests. That part of code can be used in conftest.py in case of testing globally.



```python
import pytest
from polarion_report_maker import PolarionReportMaker, Exceptions, Mode

INI_FILE = r'put path to ini file here'

###########################
##### PYTEST FIXTURES #####
###########################

@pytest.fixture(scope="session")
def polarion():
    report_maker = PolarionReportMaker(ini_path=INI_FILE,
                                       local_handler=True,
                                       mode=Mode.API.value,
                                       test_run_id=False,
                                       exceptions=Exceptions.WARNING.value)
    yield report_maker
    report_maker.create_test_run()
@pytest.fixture(scope="function",
                autouse=True)

def test_check(request, polarion: PolarionReportMaker):
    polarion.get_pytest_request(request)
    yield
    polarion.check_test_result(request)


###############################
##### TEST SESSION MODULE #####
###############################
def test_example(polarion):
    polarion.init_test_case(ID="")
    polarion.test_case_scenario()

```

- ***test_runner_template.bat*** – example of .bat file with pytest test run commands to easy run tests during automatic test session. The command is already prepared for usage it with Polarion Report Maker application

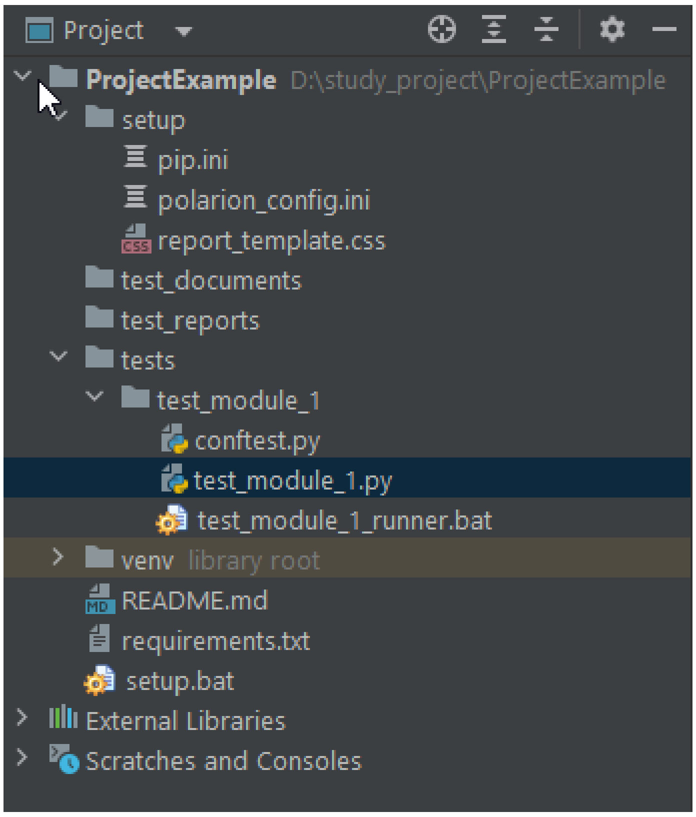


```
call ..\venv\Scripts\activate.bat
call pytest test_example.py -s --self-contained-html --css=..\report_template.css --html=..\test_reports\Test_Report.html
```

- ***report_template.css*** – template of css file to create test reports in pytest-html. Should be part of pytest test execution in command line or .bat file exec. It includes Diehl template of test report.

5. Test project structure preparation and configuration

5.1. How to prepare test project structure:



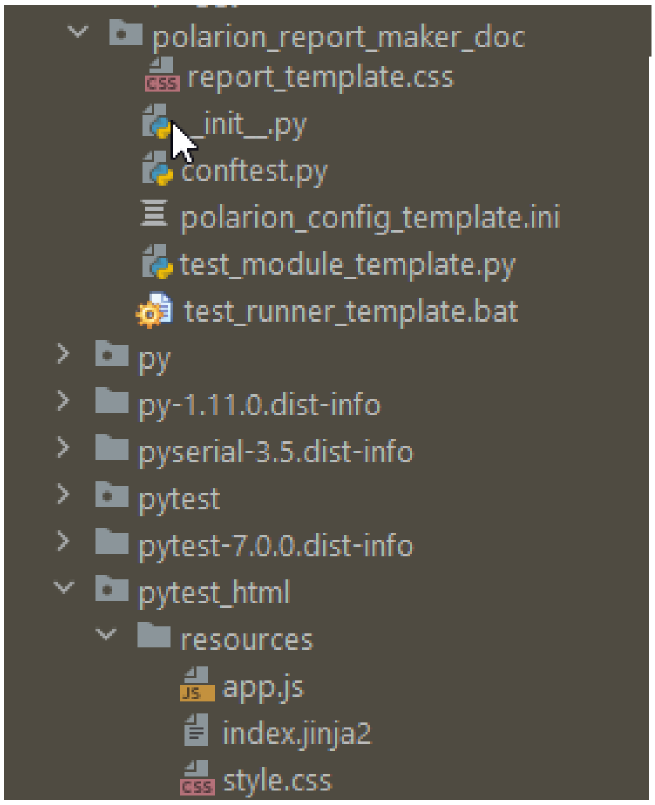
```
Project
ProjectExample  D:\study_project\ProjectExample
    setup
        pip.ini
        polarion_config.ini
        report_template.css
    test_documents
    test_reports
    tests
        test_module_1
            conftest.py
            test_module_1.py
            test_module_1_runner.bat
    venv  library root
    README.md
    requirements.txt
    setup.bat
External Libraries
Scratches and Consoles
```

- In main project file (in that case *ProjectExample*) file create files: ***tests, test_documents, test_reports***

  Note: *there are examples of file names so in your project can be used different names

`

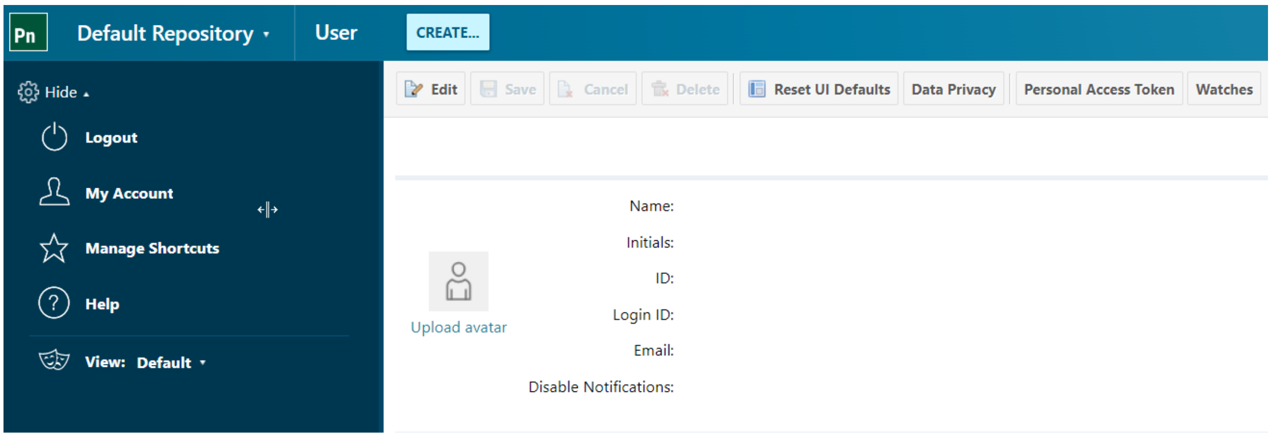
- In ***tests*** file create functional test modules and copy ***conftest.py, test_module.py and test runner.bat*** to each of created test file.
- Copy ***polarion_config.ini*** and ***report_template.css*** to ***setup*** file of main project



```
polarion_report_maker_doc
    report_template.css
    __init__.py
    conftest.py
    polarion_config_template.ini
    test_module_template.py
    test_runner_template.bat
py
py-1.11.0.dist-info
pyserial-3.5.dist-info
pytest
pytest-7.0.0.dist-info
pytest_html
    resources
        app.js
        index.jinja2
        style.css
```

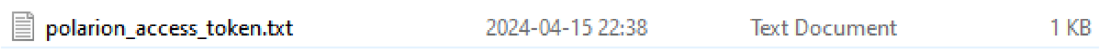
5.2. How to configure Polarion Report Maker

Create and save the Polarion Access Token from your Polarion account to have possibility of connection with internal server during tool usage. Login to Polarion and go to section *My Account*. Choose option *Personal Access Token* and generate token.



Name:
Initials:
ID:
Login ID:
Email:
Disable Notifications:

Generated token put in .txt file and save in safe location in your local disk.

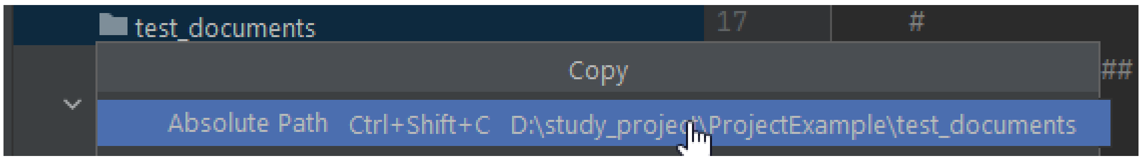


polarion_access_token.txt	2024-04-15 22:38	Text Document	1 KB

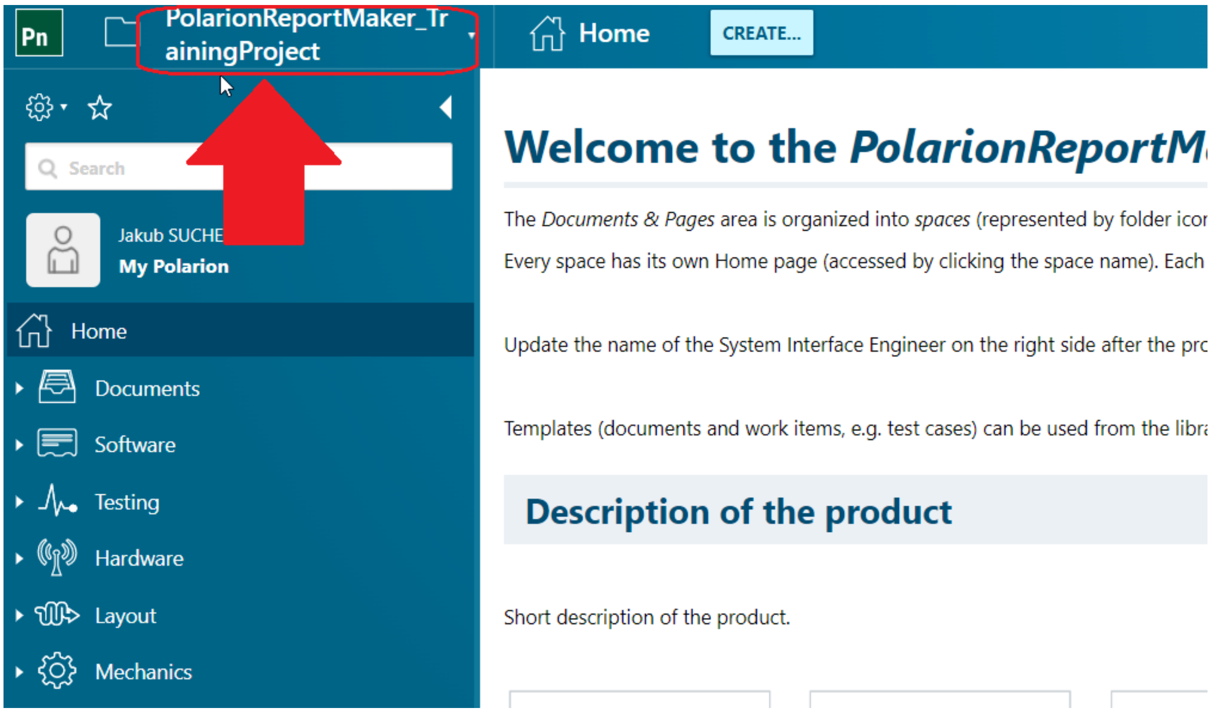
`

Configure polarion_config.ini file:

- ***POLARION_SERVER*** – that option should be not changed. This is part of API endpoint and should not be changed until new API standard
- ***TOKEN_PATH –*** put absolute path to your generated token .txt file
- ***TEST_DOCUMENT_PATH –*** put absolute path to test_documents folder in project structure (section which contains test cases in *.xlsx* exported from Polarion from Test specification)


test_documents
Copy
Absolute Path Ctrl+Shift+C D:\study_project\ProjectExample\test_documents

- ***PROJECT_ID –*** put project_id (project_name) from Polarion (see picture)



Welcome to the PolarionReportM

The *Documents & Pages* area is organized into *spaces* (represented by folder icor

Every space has its own Home page (accessed by clicking the space name). Each

Update the name of the System Interface Engineer on the right side after the prc

Templates (documents and work items, e.g. test cases) can be used from the libra

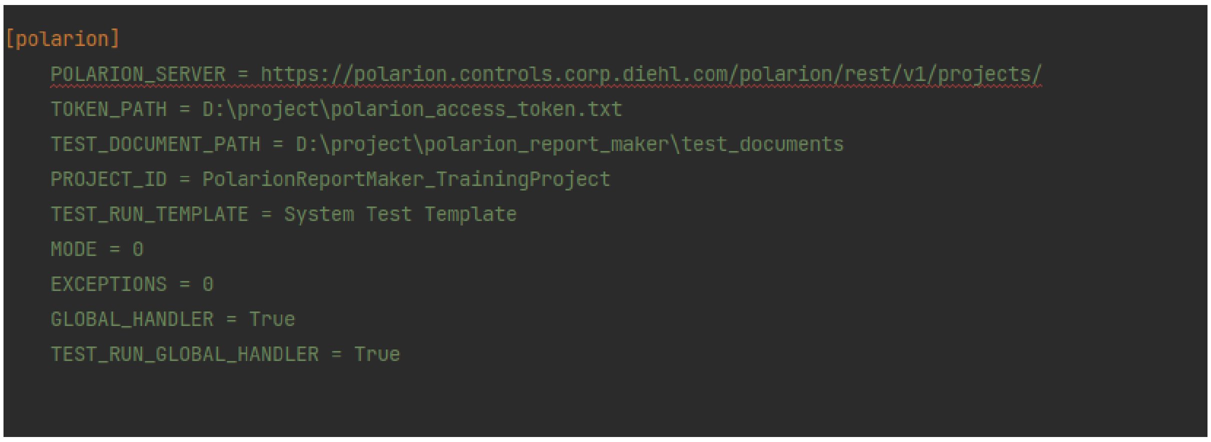
Description of the product

Short description of the product.

- ***TEST_RUN_TEMPLATE –*** Diehl template of test run layout in Polarion. Should not be changed until new/other version
- ***MODE –*** choose interface which should be used during Polarion Maker Report
  - ***0 –*** Rest API mode that download test items from Polarion server in real time (preferred during normal usage). It supports test report creation and test run uploading to Polarion server
  - ***1 -*** File mode that download test items based on exported files from Polarion before

- ***EXCEPTIONS –*** parameter that handle exceptions type during Polarion Report Maker usage
  - ***0 – Warnings level –*** issues of Polarion Report Maker tool does not have impact on test execution. Appliance report issues just after test session in log monitor
  - ***1 – Error level –*** issues of Polarion Report Maker tool have impact on test execution. That issues are treated as normal failed operation during test execution.
- ***GLOBAL_HANDLER –*** flag to global enable/disable Polarion Report Maker in project
- ***TEST_RUN_GLOBAL_HANLDER -*** flag to global enable/disable test run creation in Polarion after test execution

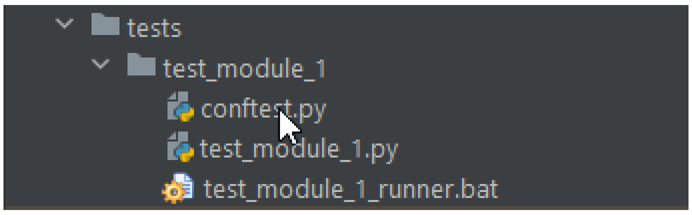
`



```
[polarion]
    POLARION_SERVER = https://polarion.controls.corp.diehl.com/polarion/rest/v1/projects/
    TOKEN_PATH = D:\project\polarion_access_token.txt
    TEST_DOCUMENT_PATH = D:\project\polarion_report_maker\test_documents
    PROJECT_ID = PolarionReportMaker_TrainingProject
    TEST_RUN_TEMPLATE = System Test Template
    MODE = 0
    EXCEPTIONS = 0
    GLOBAL_HANDLER = True
    TEST_RUN_GLOBAL_HANDLER = True
```

6. Test module configuration

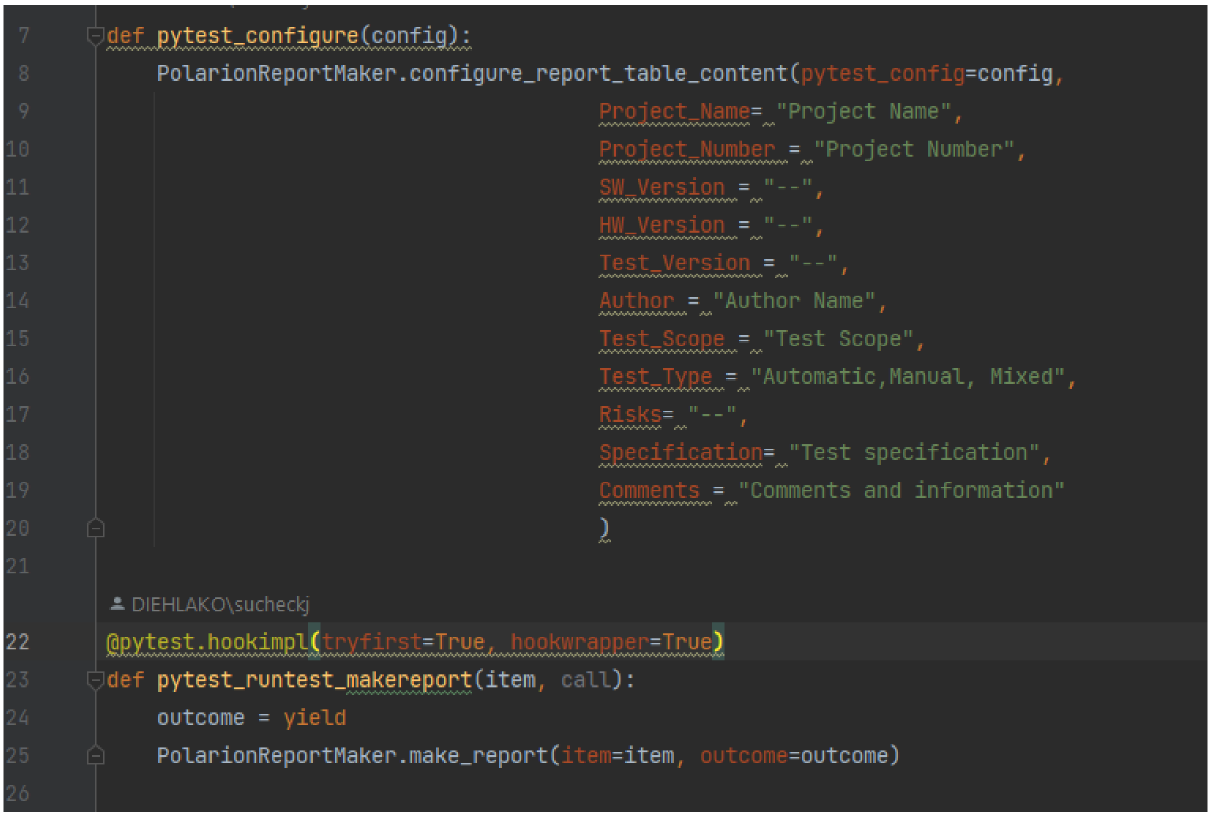
After correct configuration of polarion_config.ini file, there should be created and configured test module. The structure of that module should look like as in the picture:



```
tests
  test_module_1
    conftest.py
    test_module_1.py
    test_module_1_runner.bat
```

In ***conftest.py*** file there is important to include Pytest functions as in the picture that enable Polarion Report Maker functionality. In *pytest_configure()* you can define which section should be added to General information table in created test reports. It is important - points defined in input of *configure_report_table_content()* function will be filled in chronological way in test report. This is an example and can be adjusted to your project needs and requirements.

`

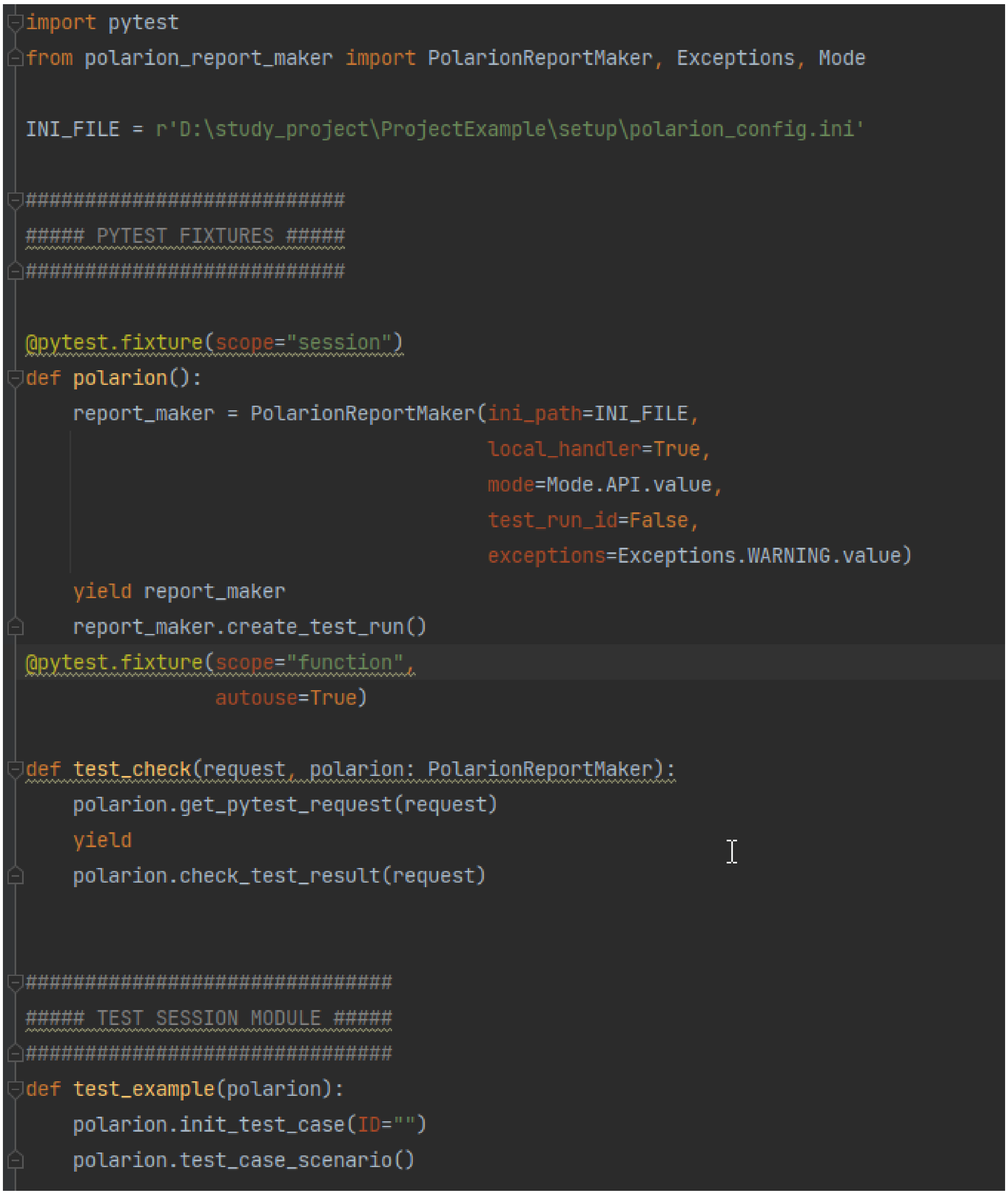


```
def pytest_configure(config):
    PolarionReportMaker.configure_report_table_content(pytest_config=config,
                                                       Project_Name= "Project Name",
                                                       Project_Number = "Project Number",
                                                       SW_Version = "--",
                                                       HW_Version = "--",
                                                       Test_Version = "--",
                                                       Author = "Author Name",
                                                       Test_Scope = "Test Scope",
                                                       Test_Type = "Automatic,Manual, Mixed",
                                                       Risks= "--",
                                                       Specification= "Test specification",
                                                       Comments = "Comments and information"
                                                       )

DIEHLAKO\sucheckj
@pytest.hookimpl(tryfirst=True, hookwrapper=True)
def pytest_runtest_makereport(item, call):
    outcome = yield
    PolarionReportMaker.make_report(item=item, outcome=outcome)
```

`

Main ***test_module.py*** of pytest framework should include structure as in the picture below. This is important to not change it to proper working of Polarion Report Maker. In that test code there is possible to put in INI_FILE variable an absolute path to polarion_config.ini before prepared in point 5. The way of insert it to PolarionReportMaker() object can be handled in different method (for example via parser).

That fixtures can be also used as global conftest fixture for all test modules. Then there fixtures should be parts of global conftest.py.



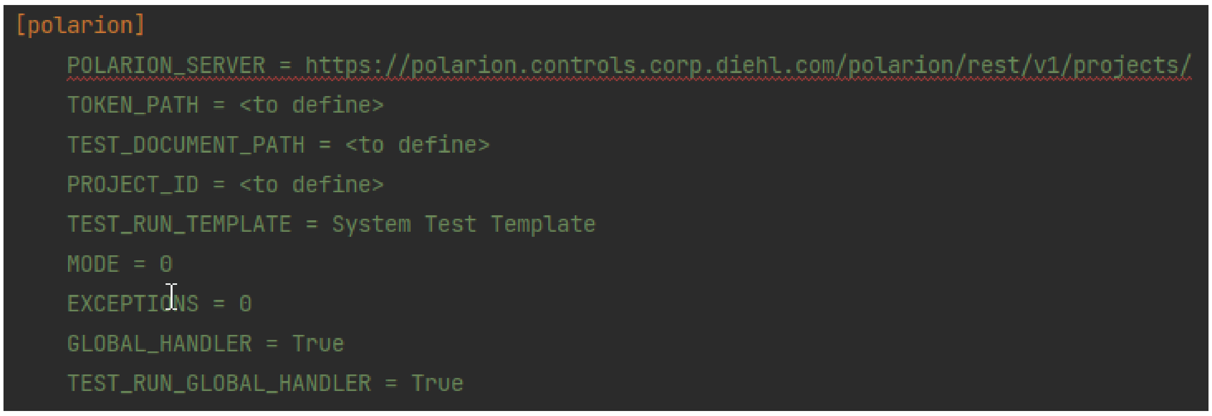
```
import pytest
from polarion_report_maker import PolarionReportMaker, Exceptions, Mode

INI_FILE = r'D:\study_project\ProjectExample\setup\polarion_config.ini'

###########################
###### PYTEST FIXTURES ######
###########################

@pytest.fixture(scope="session")
def polarion():
    report_maker = PolarionReportMaker(ini_path=INI_FILE,
                                       local_handler=True,
                                       mode=Mode.API.value,
                                       test_run_id=False,
                                       exceptions=Exceptions.WARNING.value)
    yield report_maker
    report_maker.create_test_run()
@pytest.fixture(scope="function",
                autouse=True)

def test_check(request, polarion: PolarionReportMaker):
    polarion.get_pytest_request(request)
    yield
    polarion.check_test_result(request)



###############################
###### TEST SESSION MODULE ######
###############################
def test_example(polarion):
    polarion.init_test_case(ID="")
    polarion.test_case_scenario()
```

`

7. Polarion Report Maker general usage

7.1. Global settings

Before running test sessions it is possible to configure Polarion Report Maker in two ways.
The first is global configuration. It means that these parameters will be used for all test modules in project. This configuration can be handle via polarion_config.ini:



```
[polarion]
    POLARION_SERVER = https://polarion.controls.corp.diehl.com/polarion/rest/v1/projects/
    TOKEN_PATH = <to define>
    TEST_DOCUMENT_PATH = <to define>
    PROJECT_ID = <to define>
    TEST_RUN_TEMPLATE = System Test Template
    MODE = 0
    EXCEPTIONS = 0
    GLOBAL_HANDLER = True
    TEST_RUN_GLOBAL_HANDLER = True
```

- **MODE:**
  - **True** value if PRM should be turned of globally,
  - **False** if not

- **EXCEPTIONS:**
  - **0** if Warning level should be on; Warning levels monitors issues and report it after session execution but does not have impact on passing/failing status of test cases;

  - **1** if Error level should be on; Error level totally fails executing test case

- **GLOBAL_HANDLER:**
  - **True** – Polarion Report Maker is on for whole test project
  - **False** – Polarion report Maker (test reports + test runs preparation) is off for whole project

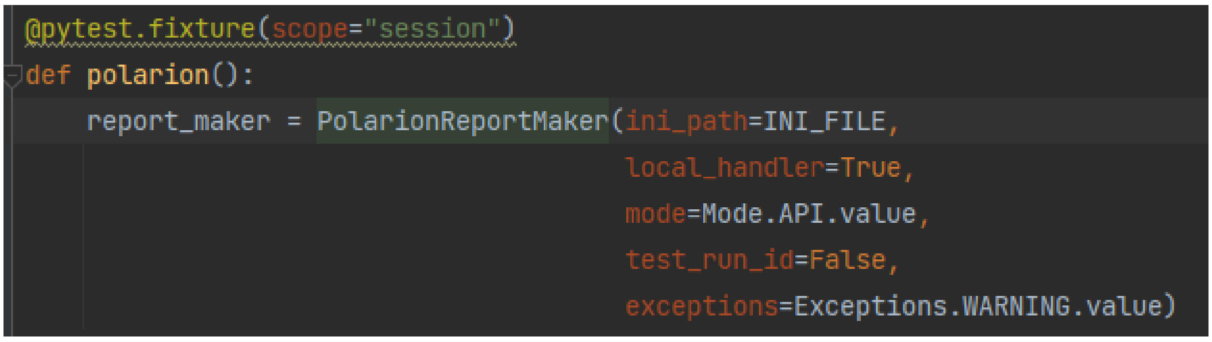
- **TEST_RUN_GLOBAL_HANDLER:**
  - **True** – Test Run uploading to Polarion is turned on for test project
  - **False** – Test Run uploading is turned off globally

Full usage of these parameters are also clarified in polarion_config.ini in comments section.

`

7.2. Local settings

Despite possible global parametrization, there is possible to configure tool only for one test module. That option is implemented due to development needs during test modules preparation , organization, and unit test of project. These options can be handled via input parameters of Polarion Maker Object in pytest.fixture.



```
@pytest.fixture(scope="session")
def polarion():
    report_maker = PolarionReportMaker(ini_path=INI_FILE,
                                       local_handler=True,
                                       mode=Mode.API.value,
                                       test_run_id=False,
                                       exceptions=Exceptions.WARNING.value)
```

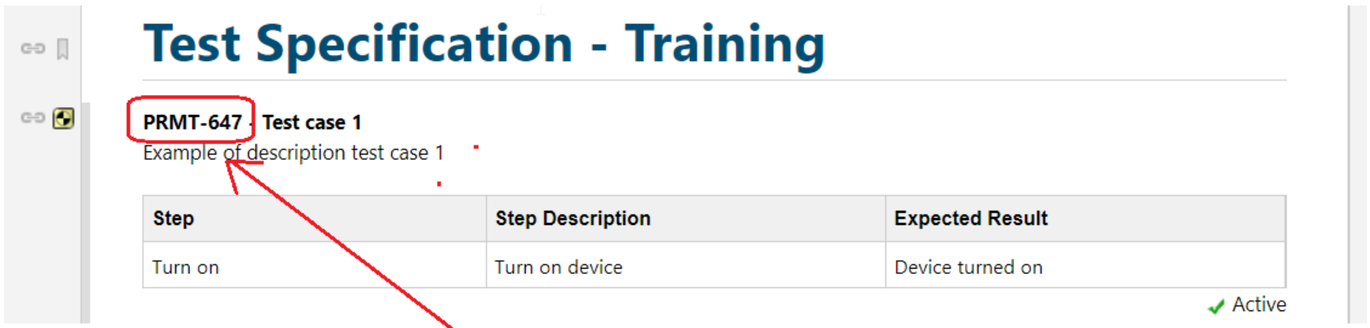
- **ini_path** – obligatory parameter which should contain string value of polarion_config.ini destination
- **local_handler** – flag to turn off locally PRM functions; ***default: <False>***
- **mode** – flag to local change of used mode; ***default API <0>***
- **test_run_id**:
  - **False, '' (empty string)** – turn off functionality of test run uploading to Polarion
  - **'test_run_name' (string with test run name) –** test run creation is enabled and test run will be named and created after test execution (if the same name not exist in Polarion yet); ***default <False>***
- **exceptions** – possibility of change exceptions level in test module; ***default Warnings <0>***

All these parameters do not have to be defined locally in function, but can be upload as parameters from external file (for example from parameter file .ini).

`

7.3. Test cases preparation

One thing which is obligatory to connect and link test cases from pytest with test cases managing in Polarion is **init_test_case(ID='')** method with correct parameter of ID taken from Polarion.



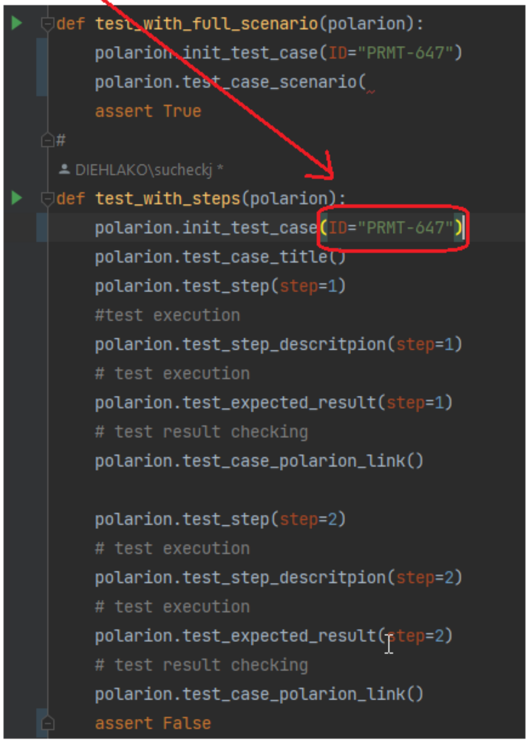
Test Specification - Training

PRMT-647 - Test case 1

Example of description test case 1

Step	Step Description	Expected Result
Turn on	Turn on device	Device turned on

✓ Active



```
def test_with_full_scenario(polarion):
    polarion.init_test_case(ID="PRMT-647")
    polarion.test_case_scenario()
    assert True
#
DIEHLAKO\sucheckj *
def test_with_steps(polarion):
    polarion.init_test_case(ID="PRMT-647")
    polarion.test_case_title()
    polarion.test_step(step=1)
    #test execution
    polarion.test_step_descritpion(step=1)
    # test execution
    polarion.test_expected_result(step=1)
    # test result checking
    polarion.test_case_polarion_link()

    polarion.test_step(step=2)
    # test execution
    polarion.test_step_descritpion(step=2)
    # test execution
    polarion.test_expected_result(step=2)
    # test result checking
    polarion.test_case_polarion_link()
    assert False
```

After correct init preparation we can use functions available in library of Polarion Report Maker:

- ***test_case_title()*** – prints title of test case from Polarion;
- ***test_step(step=1)*** – prints "Step" cell of given step number of test case
- ***test_step_description(step=1)*** – prints "Step description" cell of given step number of test case
- ***test_step-expected_result(step=1)*** - prints "Expected result" cell of given step number of test case
- ***test_case-scenario()*** – prints full scenario of initialized test case
- ***test_case_polarion_link()*** - prints polarion link of test case

If test report is generated after test execution of all functions used in test session will be as a part of test report.

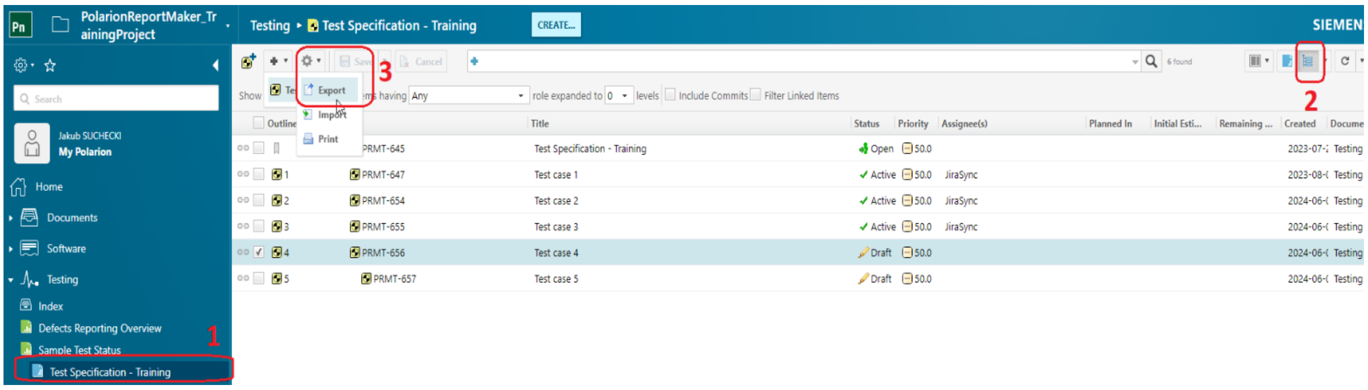
`

7.4. API Mode

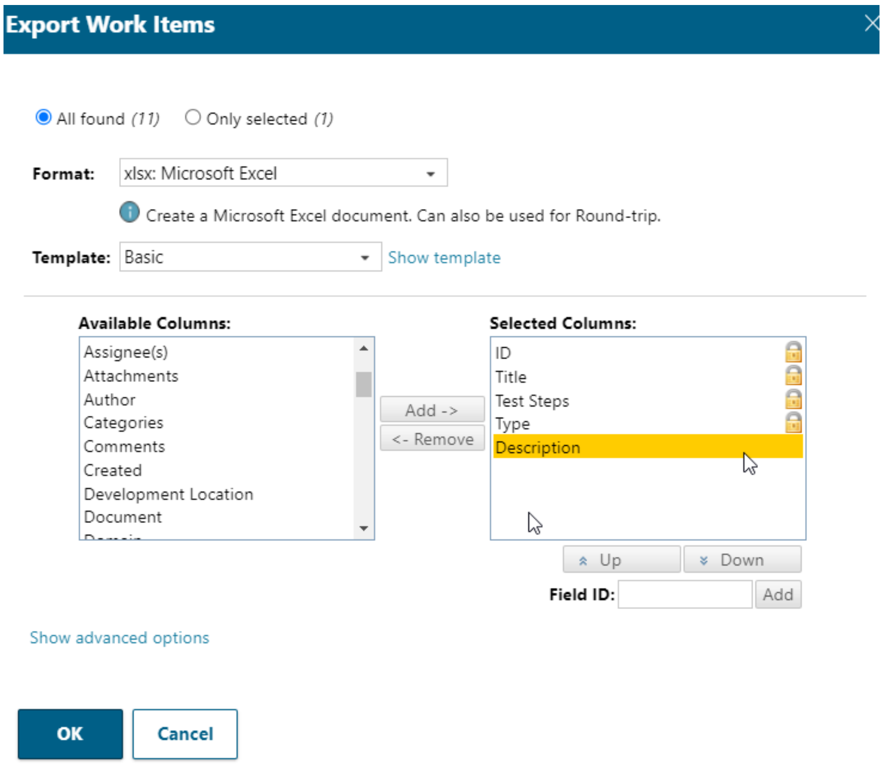
This mode handles downloading test cases data from Polarion server in real-time. This is mode prepared as default and recommended to use. Only server errors or connection errors can block using this functionality.

7.5. File Mode

File mode is prepared for working with Polarion test cases without connection with server, so that option is valid only for test report generation (is not used for test run creator). That mode can work only with test case data that are prepared from Polarion server before usage. If user wants to use file mode, firstly it is important to export full specification included in pytest project to ***test_documents*** folder. Below you can find short instructions in the pictures how to export these documents correctly.



Outline		Title	Status	Priority	Assignee(s)	Planned In	Initial Esti...	Remaining ...	Created	Docume
	PRMT-645	Test Specification - Training	Open	50.0					2023-07-:	Testing
1	PRMT-647	Test case 1	✓ Active	50.0	JiraSync				2023-08-(	Testing
2	PRMT-654	Test case 2	✓ Active	50.0	JiraSync				2024-06-(	Testing
3	PRMT-655	Test case 3	✓ Active	50.0	JiraSync				2024-06-(	Testing
✓ 4	PRMT-656	Test case 4	Draft	50.0					2024-06-(	Testing
5	PRMT-657	Test case 5	Draft	50.0					2024-06-(	Testing



Export Work Items

(●) All found *(11)* ( ) Only selected *(1)*

Format: xlsx: Microsoft Excel

Create a Microsoft Excel document. Can also be used for Round-trip.

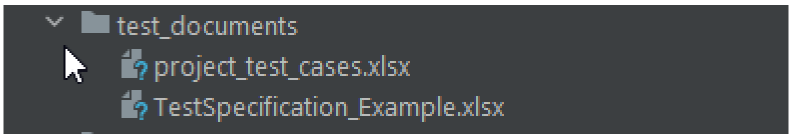
Template: Basic Show template

Available Columns:
Assignee(s)
Attachments
Author
Categories
Comments
Created
Development Location
Document

Selected Columns:
ID
Title
Test Steps
Type
Description

`

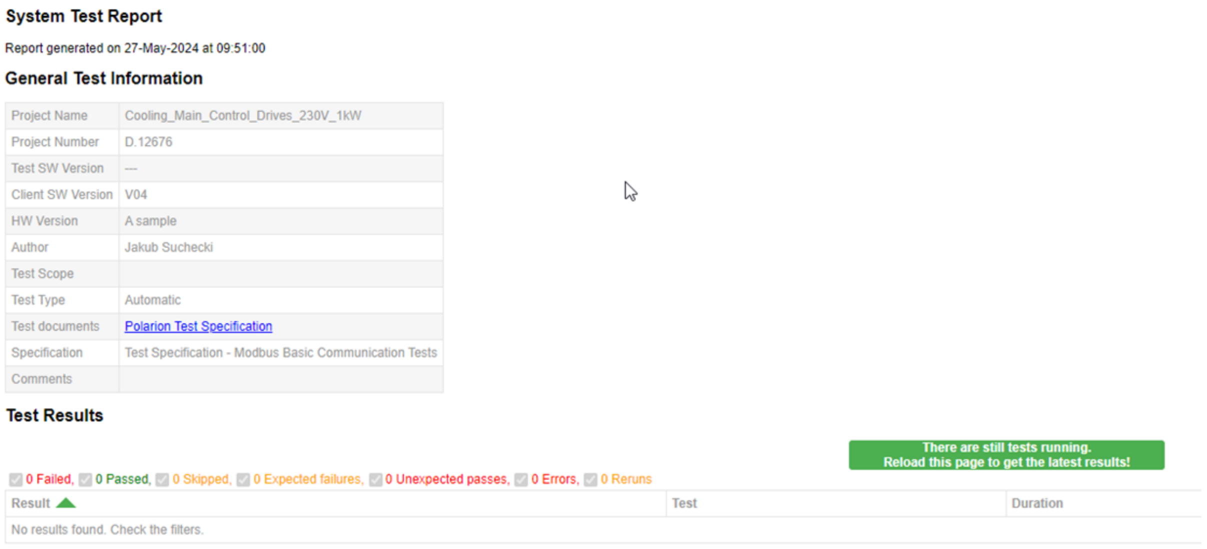
All exported documents should be uploaded to ***test_documents*** folder in project files:



test_documents
project_test_cases.xlsx
TestSpecification_Example.xlsx

Thanks to these steps and method PRM appliance uses test cases data directly from exported files.

8. Test report example



System Test Report

Report generated on 27-May-2024 at 09:51:00

General Test Information

Project Name	Cooling_Main_Control_Drives_230V_1kW
Project Number	D.12676
Test SW Version	—
Client SW Version	V04
HW Version	A sample
Author	Jakub Suchecki
Test Scope	
Test Type	Automatic
Test documents	Polarion Test Specification
Specification	Test Specification - Modbus Basic Communication Tests
Comments	

Test Results

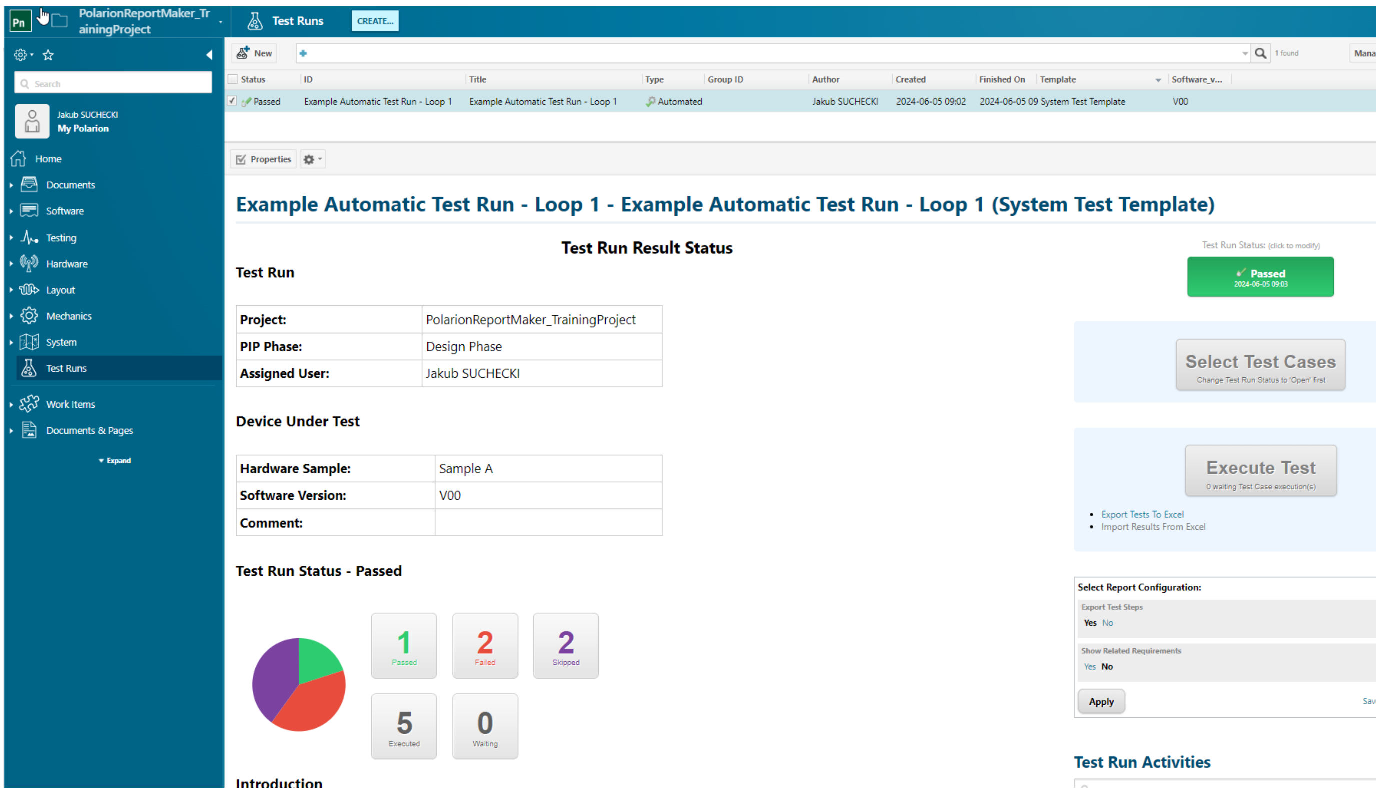
There are still tests running.
Reload this page to get the latest results!

✓ 0 Failed, ✓ 0 Passed, ✓ 0 Skipped, ✓ 0 Expected failures, ✓ 0 Unexpected passes, ✓ 0 Errors, ✓ 0 Reruns

Result	Test	Duration
No results found. Check the filters.		

`

9. Test run example



Status	ID	Title	Type	Group ID	Author	Created	Finished On	Template	Software_v...
Passed	Example Automatic Test Run - Loop 1	Example Automatic Test Run - Loop 1	Automated		Jakub SUCHECKI	2024-06-05 09:02	2024-06-05 09	System Test Template	V00

Example Automatic Test Run - Loop 1 - Example Automatic Test Run - Loop 1 (System Test Template)

Test Run Result Status

Test Run

Project:	PolarionReportMaker_TrainingProject
PIP Phase:	Design Phase
Assigned User:	Jakub SUCHECKI

Device Under Test

Hardware Sample:	Sample A
Software Version:	V00
Comment:	

Test Run Status - Passed

1 Passed
2 Failed
2 Skipped
5 Executed
0 Waiting

Introduction

Test Run Status: (click to modify)
Passed 2024-06-05 09:03

Select Test Cases
Change Test Run Status to 'Open' first

Execute Test
0 waiting Test Case execution(s)

- Export Tests To Excel
- Import Results From Excel

Select Report Configuration:

Export Test Steps
Yes No

Show Related Requirements
Yes **No**

Apply

Test Run Activities