PyTestLog2DB

v. 0.1.1

Tran Duy Ngoan

22.11.2022

Contents

Chapter 1

Introduction

TODO

Chapter 2

Description

2.1 Get the pytest XML result

In order to import the execution result(s), the ***.xml** file which contains result of all executed pytest testcases is required.

But that file is not generated by default. The argument --junit-xml=<log> needs to be specified when executing the pytest to get the generated JUnit XML result file at given path.

The example pytest command to get the ***.xml** result file: :

```
pytest --junit-xml=path/to/result.xml pytest/folder
```

2.2 Tool features

2.2.1 Usage

Use below command to get tools's usage :

```
PyTestLog2DB -h
```

The usage should be showed as below: :

```
usage: PyTestLog2DB (PyTestXMLReport to TestResultWebApp importer) [-h] [-v]
                [--recursive] [--dryrun] [--append] [--UUID UUID]
                [--config CONFIG] resultxmlfile server user password database

PyTestLog2DB imports pytest JUnit XML report file(s)generated by pytest into
                a WebApp database.

positional arguments:
resultxmlfile    absolute or relative path to the pytest JUnit XML report
                file or directory of report files to be imported.
server           server which hosts the database (IP or URL).
user             user for database login.
password         password for database login.
database         database schema for database login.

optional arguments:
-h, --help       show this help message and exit
-v               Version of the PyTestLog2DB importer.
--recursive      if set, then the path is searched recursively for output
                files to be imported.
--dryrun         if set, then verify all input arguments (includes DB connection)
                and show what would be done.
--append         is used in combination with -UUID <UUID>. If set, allow to append
```

```
                  new result(s) to existing execution result UUID in -UUID argument.
--UUID UUID       UUID used to identify the import and version ID on webapp.
                  If not provided PyTestLog2DB will generate an UUID for the whole import.
--config CONFIG   configuration json file for component mapping information.
```

The below command is simple usage with all required arguments to import PyTest results into TestResultWebApp's database: :

```
PyTestLog2DB <resultxmlfile> <server> <user> <password> <database>
```

Besides the executable file, you can also run tool as a Python module :

```
python -m PyTestLog2DB <resultxmlfile> <server> <user> <password> <database>
```

2.2.2 Verify provided arguments

Sometimes, we just want to validate the ***.xml** and database connection without changing anything in the database, the optional argument --dryrun can be used in this case.

When executing in dryrun mode, PyTestLog2DB will:

- Verify the provided ***.xml** file is valid or not.
- Verify the database connection with provided credential.
- Verify other information which given in optional arguments.
- Just print all test cases will be imported without touching database.

This feature will helps you to ensure that there is no error when executing PyTestLog2DB tool (normal mode)to create new record(s) and update TestResultWebApp's database.

2.2.3 Searching *.xml result file(s)

TODO

2.2.4 Handle missing information

The ***.xml** report file which is generated by PyTest contains only the testcase result(s) and less metadata information about the test execution such as *project/variant*, *software version*, *tester* , *component*, ... which are required by TestResultWebApp.

So that, PyTestLog2DB will handle those information with the default values as below:

- *project/variant* : **PyTest**
- *version_sw* : execution time as *%Y%m%d_%H%M%S* format. E.g **20221128_143547**
- *version_hw* : **""**
- *version_test* : **""**
- *component* : **unknown**
- *testtool* : current python and pytest version. E.g **PyTest 6.2.5 (Python 3.9.0)**
- *tester* : current user.

However, those information can be specified in the configuration json file with option argument --config CONFIG when executing import command.

Required type for those information is **string** except the *component*. Type of *component* info can be:

- **string**: to specify the same *component* for all testcase within this execution.

- **object**: to specify the mapping between *component* info and *classname* of testcase.

Sample configuration json file: :

```
{
   "variant"      : "MyProject",
   "version_sw"   : "0.1.1",
   "component"    : {
      "Testsuite1"      : "test-data.test_tsclass.TestSuite1",
      "Testsuite2"      : "test-data.test_tsclass.TestSuite2",
      "Others"          : [
         "test-data.test_ts1",
         "test-data.test_ts2"
      ]
   },
   "tester"       : "Tran Duy Ngoan"
}
```

As above sample configuration, the component mapping can be explained as below:

- Testcase(s) with classname **test-data.test_tsclass.TestSuite1** is belong to component **Testsuite1**
- Testcase(s) with classname **test-data.test_tsclass.TestSuite2** is belong to component **Testsuite2**
- And component **Others** contains all testcases with classnames **test-data.test_ts1** and **test-data.test_ts2**.

With this feature, the importing execution result can be displayed properly without missing any required information.

2.2.5 Append mode

TODO

2.3 Display on TestResultWebApp

TODO

Chapter 3

CDataBase.py

3.1 Class: CDataBase

Imported by:

```
from PyTestLog2DB.CDataBase import CDataBase
```

CDataBase class play a role as mysqlclient and provide methods to interact with TestResultWebApp's database.

3.1.1 Method: connect

Connect to the database with provided authentication and db info.

Arguments:

- `host`
  / *Condition*: required / *Type*: str /
  URL which is hosted the TestResultWebApp's database.
- `user`
  / *Condition*: required / *Type*: str /
  User name for database authentication.
- `passwd`
  / *Condition*: required / *Type*: str /
  User's password for database authentication.
- `database`
  / *Condition*: required / *Type*: str /
  Database name.
- `charset`
  / *Condition*: optional / *Type*: str / *Default*: 'utf8' /
  The connection character set.
- `use_unicode`
  / *Condition*: optional / *Type*: bool / *Default*: True /
  If True, CHAR and VARCHAR and TEXT columns are returned as Unicode strings, using the configured character set.

Returns:

(*no returns*)

3.1.2 Method: disconnect

Disconnect from TestResultWebApp's database.

Arguments:

(*no arguments*)

Returns:

(*no returns*)

3.1.3 Method: cleanAllTables

Delete all table data. Please be careful before calling this method.

Arguments:

(*no arguments*)

Returns:

(*no returns*)

3.1.4 Method: sCreateNewTestResult

Creates a new test result in `tbl_result`. This is the main table which is linked to all other data by means of `test_result_id`.

Arguments:

- `_tbl_prj_project`
  / *Condition*: required / *Type*: str /
  Project information.
- `_tbl_prj_variant`
  / *Condition*: required / *Type*: str /
  Variant information.
- `_tbl_prj_branch`
  / *Condition*: required / *Type*: str /
  Branch information.
- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result.
- `_tbl_result_interpretation`
  / *Condition*: required / *Type*: str /
  Result interpretation.
- `_tbl_result_time_start`
  / *Condition*: required / *Type*: str /
  Test result start time as format `%Y-%m-%d %H:%M:%S`.
- `_tbl_result_time_end`
  / *Condition*: required / *Type*: str /
  Test result end time as format `%Y-%m-%d %H:%M:%S`.
- `_tbl_result_version_sw_target`
  / *Condition*: required / *Type*: str /
  Software version information.

- _tbl_result_version_sw_test
  / *Condition*: required / *Type*: str /
  Test version information.
- _tbl_result_version_target
  / *Condition*: required / *Type*: str /
  Hardware version information.
- _tbl_result_jenkinsurl
  / *Condition*: required / *Type*: str /
  Jenkinsurl in case test result is executed by jenkins.
- _tbl_result_reporting_qualitygate
  / *Condition*: required / *Type*: str /
  Qualitygate information for reporting.

Returns:

- _tbl_test_result_id
  / *Type*: str /
  test_result_id of new test result.

3.1.5 Method: nCreateNewFile

Create new file entry in tbl_file table.

Arguments:

- _tbl_file_name
  / *Condition*: required / *Type*: str /
  File name information.
- _tbl_file_tester_account
  / *Condition*: required / *Type*: str /
  Tester account information.
- _tbl_file_tester_machine
  / *Condition*: required / *Type*: str /
  Test machine information.
- _tbl_file_time_start
  / *Condition*: required / *Type*: str /
  Test file start time as format %Y-%m-%d %H:%M:%S.
- _tbl_file_time_end
  / *Condition*: required / *Type*: str /
  Test file end time as format %Y-%m-%d %H:%M:%S.
- _tbl_test_result_id
  / *Condition*: required / *Type*: str /
  UUID of test result for linking to tbl_result table.
- _tbl_file_origin
  / *Condition*: required / *Type*: str /
  Origin (test framework) of test file. Deafult is "ROBFW"

Returns:

- iInsertedID
  / *Type*: int /
  ID of new entry.

3.1.6 Method: vCreateNewHeader

Create a new header entry in `tbl_file_header` table which is linked with the file.

Arguments:

- `_tbl_file_id`
  / *Condition*: required / *Type*: int /
  File ID information.
- `_tbl_header_testtoolconfiguration_testtoolname`
  / *Condition*: required / *Type*: str /
  Test tool name.
- `_tbl_header_testtoolconfiguration_testtoolversionstring`
  / *Condition*: required / *Type*: str /
  Test tool version.
- `_tbl_header_testtoolconfiguration_projectname`
  / *Condition*: required / *Type*: str /
  Project name.
- `_tbl_header_testtoolconfiguration_logfileencoding`
  / *Condition*: required / *Type*: str /
  Encoding of logfile.
- `_tbl_header_testtoolconfiguration_pythonversion`
  / *Condition*: required / *Type*: str /
  Python version info.
- `_tbl_header_testtoolconfiguration_testfile`
  / *Condition*: required / *Type*: str /
  Test file name.
- `_tbl_header_testtoolconfiguration_logfilepath`
  / *Condition*: required / *Type*: str /
  Path to log file.
- `_tbl_header_testtoolconfiguration_logfilemode`
  / *Condition*: required / *Type*: str /
  Mode of log file.
- `_tbl_header_testtoolconfiguration_ctrlfilepath`
  / *Condition*: required / *Type*: str /
  Path to control file.
- `_tbl_header_testtoolconfiguration_configfile`
  / *Condition*: required / *Type*: str /
  Path to configuration file.
- `_tbl_header_testtoolconfiguration_confname`
  / *Condition*: required / *Type*: str /
  Configuration name.
- `_tbl_header_testfileheader_author`
  / *Condition*: required / *Type*: str /
  File author.

- _tbl_header_testfileheader_project
  / *Condition*: required / *Type*: str /
  Project information.
- _tbl_header_testfileheader_testfiledate
  / *Condition*: required / *Type*: str /
  File creation date.
- _tbl_header_testfileheader_version_major
  / *Condition*: required / *Type*: str /
  File major version.
- _tbl_header_testfileheader_version_minor
  / *Condition*: required / *Type*: str /
  File minor version.
- _tbl_header_testfileheader_version_patch
  / *Condition*: required / *Type*: str /
  File patch version.
- _tbl_header_testfileheader_keyword
  / *Condition*: required / *Type*: str /
  File keyword.
- _tbl_header_testfileheader_shortdescription
  / *Condition*: required / *Type*: str /
  File short description.
- _tbl_header_testexecution_useraccount
  / *Condition*: required / *Type*: str /
  Tester account who run the execution.
- _tbl_header_testexecution_computername
  / *Condition*: required / *Type*: str /
  Machine name which is executed on.
- _tbl_header_testrequirements_documentmanagement
  / *Condition*: required / *Type*: str /
  Requirement management information.
- _tbl_header_testrequirements_testenvironment
  / *Condition*: required / *Type*: str /
  Requirement environment information.
- _tbl_header_testbenchconfig_name
  / *Condition*: required / *Type*: str /
  Testbench configuration name.
- _tbl_header_testbenchconfig_data
  / *Condition*: required / *Type*: str /
  Testbench configuration data.
- _tbl_header_preprocessor_filter
  / *Condition*: required / *Type*: str /
  Preprocessor filter information.
- _tbl_header_preprocessor_parameters
  / *Condition*: required / *Type*: str /
  Preprocessor parameters definition.

Returns:

(*no returns*)

3.1.7 Method: nCreateNewSingleTestCase

Create single testcase entry in `tbl_case` table immediately.

Arguments:

- `_tbl_case_name`
  / *Condition*: required / *Type*: str /
  Test case name.
- `_tbl_case_issue`
  / *Condition*: required / *Type*: str /
  Test case issue ID.
- `_tbl_case_tcid`
  / *Condition*: required / *Type*: str /
  Test case ID (used for testmanagement tool).
- `_tbl_case_fid`
  / *Condition*: required / *Type*: str /
  Test case requirement (function) ID.
- `_tbl_case_testnumber`
  / *Condition*: required / *Type*: int /
  Order of test case in file.
- `_tbl_case_repeatcount`
  / *Condition*: required / *Type*: int /
  Test case repeatition count.
- `_tbl_case_component`
  / *Condition*: required / *Type*: str /
  Component which test case is belong to.
- `_tbl_case_time_start`
  / *Condition*: required / *Type*: str /
  Test case start time as format `%Y-%m-%d %H:%M:%S`.
- `_tbl_case_result_main`
  / *Condition*: required / *Type*: str /
  Test case main result.
- `_tbl_case_result_state`
  / *Condition*: required / *Type*: str /
  Test case completion state.
- `_tbl_case_result_return`
  / *Condition*: required / *Type*: int /
  Test case result code (as integer).
- `_tbl_case_counter_resets`
  / *Condition*: required / *Type*: int /
  Counter of target reset within test case execution.
- `_tbl_case_lastlog`
  / *Condition*: required / *Type*: str /
  Traceback information when test case is failed.

- _tbl_test_result_id
  / *Condition*: required / *Type*: str /
  UUID of test result for linking to file in tbl_result table.
- _tbl_file_id
  / *Condition*: required / *Type*: int /
  Test file ID for linking to file in tbl_file table.

Returns:

- iInsertedID
  / *Type*: int /
  ID of new entry.

3.1.8 Method: nCreateNewTestCase

Create bulk of test case entries: new test cases are buffered and inserted as bulk.
Once __NUM_BUFFERD_ELEMENTS_FOR_EXECUTEMANY is reached, the creation query is executed.
Arguments:

- _tbl_case_name
  / *Condition*: required / *Type*: str /
  Test case name.
- _tbl_case_issue
  / *Condition*: required / *Type*: str /
  Test case issue ID.
- _tbl_case_tcid
  / *Condition*: required / *Type*: str /
  Test case ID (used for testmanagement tool).
- _tbl_case_fid
  / *Condition*: required / *Type*: str /
  Test case requirement (function) ID.
- _tbl_case_testnumber
  / *Condition*: required / *Type*: int /
  Order of test case in file.
- _tbl_case_repeatcount
  / *Condition*: required / *Type*: int /
  Test case repeatition count.
- _tbl_case_component
  / *Condition*: required / *Type*: str /
  Component which test case is belong to.
- _tbl_case_time_start
  / *Condition*: required / *Type*: str /
  Test case start time as format %Y-%m-%d %H:%M:%S.
- _tbl_case_result_main
  / *Condition*: required / *Type*: str /
  Test case main result.

- _tbl_case_result_state
  / *Condition*: required / *Type*: str /
  Test case completion state.
- _tbl_case_result_return
  / *Condition*: required / *Type*: int /
  Test case result code (as integer).
- _tbl_case_counter_resets
  / *Condition*: required / *Type*: int /
  Counter of target reset within test case execution.
- _tbl_case_lastlog
  / *Condition*: required / *Type*: str /
  Traceback information when test case is failed.
- _tbl_test_result_id
  / *Condition*: required / *Type*: str /
  UUID of test result for linking to file in tbl_result table.
- _tbl_file_id
  / *Condition*: required / *Type*: int /
  Test file ID for linking to file in `tbl_file` table.

Returns:
(*no returns*)

3.1.9 Method: vCreateTags

Create tag entries.
Arguments:

- _tbl_test_result_id
  / *Condition*: required / *Type*: str /
  UUID of test result.
- _tbl_usr_result_tags
  / *Condition*: required / *Type*: str /
  User tags information.

Returns:
(*no returns*)

3.1.10 Method: vSetCategory

Create category entry.
Arguments:

- _tbl_test_result_id
  / *Condition*: required / *Type*: str /
  UUID of test result.
- tbl_result_category_main
  / *Condition*: required / *Type*: str /
  Category information.

Returns:
(*no returns*)

3.1.11 Method: vUpdateStartEndTime

Create start-end time entry.

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result.
- `_tbl_result_time_start`
  / *Condition*: required / *Type*: str /
  Result start time as format `%Y-%m-%d %H:%M:%S`.
- `_tbl_result_time_end`
  / *Condition*: required / *Type*: str /
  Result end time as format `%Y-%m-%d %H:%M:%S`.

Returns:

(*no returns*)

3.1.12 Method: arGetCategories

Get existing categories.

Arguments:

(*no arguments*)

Returns:

- `arCategories`
  / *Type*: list /
  List of exsiting categories.

3.1.13 Method: vCreateAbortReason

Create abort reason entry.

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result.
- `_tbl_abort_reason`
  / *Condition*: required / *Type*: str /
  Abort reason.
- `_tbl_abort_message`
  / *Condition*: required / *Type*: str /
  Detail message of abort.

Returns:

(*no returns*)

3.1.14 Method: vCreateReanimation

Create reanimation entry.

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result.
- `_tbl_num_of_reanimation`
  / *Condition*: required / *Type*: int /
  Counter of target reanimation during execution.

Returns:

(*no returns*)

3.1.15 Method: vCreateCCRdata

Create CCR data per test case.

Arguments:

- `_tbl_test_case_id`
  / *Condition*: required / *Type*: int /
  test case ID.
- `lCCRdata`
  / *Condition*: required / *Type*: list /
  list of CCR data.

Returns:

(*no returns*)

3.1.16 Method: vFinishTestResult

Finish upload:

- First do bulk insert of rest of test cases if buffer is not empty.
- Then set state to "new report".

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result.

Returns:

(*no returns*)

3.1.17 Method: vUpdateEvtbls

Call `update_evtbls` stored procedure.

Arguments:

(*no arguments*)

Returns:

(*no returns*)

3.1.18 Method: vUpdateEvtbl

Call `update_evtbl` stored procedure to update provided `test_result_id`.

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result.

Returns:

(*no returns*)

3.1.19 Method: vEnableForeignKeyCheck

Switch `foreign_key_checks` flag.

Arguments:

- `enable`
  / *Condition*: optional / *Type*: bool / *Default*: True /
  If True, enable foreign key constraint.

Returns:

(*no returns*)

3.1.20 Method: sGetLatestFileID

Get latest file ID from `tbl_file` table.

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result.

Returns:

- `_tbl_file_id`
  / *Type*: int /
  File ID.

3.1.21 Method: vUpdateFileEndTime

Update test file end time.

Arguments:

- `_tbl_file_id`
  / *Condition*: required / *Type*: int /
  File ID to be updated.
- `_tbl_file_time_end`
  / *Condition*: required / *Type*: str /
  File end time as format `%Y-%m-%d %H:%M:%S`.

Returns:

(*no returns*)

3.1.22 Method: vUpdateResultEndTime

Update test result end time.

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: int /
  Result UUID to be updated.
- `_tbl_result_time_end`
  / *Condition*: required / *Type*: str /
  Result end time as format `%Y-%m-%d %H:%M:%S`.

Returns:

(*no returns*)

3.1.23 Method: bExistingResultID

Verify the given test result UUID is existing in `tbl_result` table or not.

Arguments:

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: int /
  Result UUID to be verified.

Returns:

- `bExisting`
  / *Type*: bool /
  True if test result UUID is already existing.

Chapter 4

pytestlog2db.py

4.1 Function: is_valid_uuid

Verify the given UUID is valid or not.

Arguments:

- `uuid_to_test`
  / *Condition*: required / *Type*: str /
  UUID to be verified.
- `version`
  / *Condition*: optional / *Type*: int / *Default*: 4 /
  UUID version.

Returns:

- `bValid`
  / *Type*: bool /
  True if the given UUID is valid.

4.2 Function: is_valid_config

Validate the json configuration base on given schema.

Default schema supports below information:

```
CONFIG_SCHEMA = {
    "component" : [str, dict],
    "variant"   : str,
    "version_sw": str,
    "version_hw": str,
    "version_test": str,
    "testtool"  :  str,
    "tester"    :  str
}
```

Arguments:

- `dConfig`
  / *Condition*: required / *Type*: dict /
  Json configuration object to be verified.

- `dSchema`
  / *Condition*: optional / *Type*: dict / *Default*: CONFIG_SCHEMA /
  Schema for the validation.
- `bExitOnFail`
  / *Condition*: optional / *Type*: bool / *Default*: True /
  If True, exit tool in case the validation is fail.

Returns:

- `bValid`
  / *Type*: bool /
  True if the given json configuration data is valid.

4.3 Function: validate_db_str_field

Validate the string value for database field bases on its acceptable length.The error will be thrown and tool terminates if the verification is failed.

Arguments:

- `field`
  / *Condition*: required / *Type*: str /
  Field name in the database.
- `value`
  / *Condition*: required / *Type*: str /
  String value to be verified.

Returns:

/ *Type*: str /
String value if the verification is fine.

4.4 Function: truncate_db_str_field

Truncate input string before importing to database.

Arguments:

- `sString`
  / *Condition*: required / *Type*: str /
  Input string for truncation.
- `iMaxLength`
  / *Condition*: required / *Type*: int /
  Max length of string to be allowed.
- `sEndChars`
  / *Condition*: optional / *Type*: str / *Default*: "..." /
  End characters which are added to end of truncated string.

Returns:

- `content`
  / *Type*: str /
  String after truncation.

4.5 Function: parse_pytest_xml

Parse and merge all given pytest *.xml result files into one result file.Besides, starttime and endtime are also calculated and added in the merged result.

Arguments:

- `xmlfiles`
  / *Condition*: required / *Type*: str /
  Path to pytest *.xml result file(s).

Returns:

- `oMergedTree`
  / *Type*: etree._Element object /
  The result object which is parsed from provided pytest *.xml result file(s).

4.6 Function: get_branch_from_swversion

Get branch name from software version information.

Convention of branch information in suffix of software version:

- All software version with .0F is the main/freature branch. The leading number is the current year. E.g. `17.0F03`
- All software version with `.1S`, `.2S`, ... is a stabi branch. The leading number is the year of branching out for stabilization. The number before "S" is the order of branching out in the year.

Arguments:

- `sw_version`
  / *Condition*: required / *Type*: str /
  Software version.

Returns:

- `branch_name`
  / *Type*: str /
  Branch name.

4.7 Function: get_test_result

Get test result from provided Testcase object.

Arguments:

- `oTest`
  / *Condition*: required / *Type*: etree._Element object /
  Testcase object.

Returns:

/ *Type*: typle /
Testcase result which contains result_main, lastlog and result_return.

4.8 Function: process_component_info

Return the component name bases on provided testcase's classname and component mapping.

Arguments:

- `dConfig`
  / *Condition*: required / *Type*: dict /
  Configuration which contains the mapping between component and testcase's classname.
- `sTestClassname`
  / *Condition*: required / *Type*: str /
  Testcase's classname to get the component info.

Returns:

- `sComponent`
  / *Type*: typle /
  Component name maps with given testcase's classname. Otherwise, "unknown" will be return as component name.

4.9 Function: process_config_file

Parse information from configuration file:

- `component`:

```
{
   "component" : {
      "componentA" : "componentA/path/to/testcase",
      "componentB" : "componentB/path/to/testcase",
      "componentC" : [
         "componentC1/path/to/testcase",
         "componentC2/path/to/testcase"
      ]
   }
}
```

  Then all testcases which their paths contain `componentA/path/to/testcase` will be belong to `componentA`, ...

Arguments:

- `config_file`
  / *Condition*: required / *Type*: str /
  Path to configuration file.

Returns:

- `dConfig`
  / *Type*: dict /
  Configuration object.

4.10 Function: process_test

Process test case data and create new test case record.

Arguments:

- `db`
  / *Condition*: required / *Type*: CDataBase object /
  CDataBase object.
- `test`
  / *Condition*: required / *Type*: etree._Element object /
  Robot test object.
- `file_id`
  / *Condition*: required / *Type*: int /
  File ID for mapping.
- `test_result_id`
  / *Condition*: required / *Type*: str /
  Test result ID for mapping.
- `component_name`
  / *Condition*: required / *Type*: str /
  Component name which this test case is belong to.
- `test_number`
  / *Condition*: required / *Type*: int /
  Order of test case in file.
- `start_time`
  / *Condition*: required / *Type*: datetime object /
  Start time of testcase.

Returns:

/ *Type*: float /
Duration (in second) of test execution.

4.11 Function: process_suite

Process to the lowest suite level (test file):

- Create new file and its header information
- Then, process all child test cases

Arguments:

- `db`
  / *Condition*: required / *Type*: CDataBase object /
  CDataBase object.
- `suite`
  / *Condition*: required / *Type*: etree._Element object /
  Robot suite object.

- `_tbl_test_result_id`
  / *Condition*: required / *Type*: str /
  UUID of test result for importing.
- `dConfig`
  / *Condition*: required / *Type*: dict / *Default*: None /
  Configuration data which is parsed from given json configuration file.

Returns:
(*no returns*)

4.12 Function: PyTestLog2DB

Import pytest results from `*.xml` file(s) to TestResultWebApp's database.
Flow to import PyTest results to database:

1. Process provided arguments from command line.
2. Parse PyTest results.
3. Connect to database.
4. Import results into database.
5. Disconnect from database.

Arguments:

- `args`
  / *Condition*: required / *Type*: ArgumentParser object /
  Argument parser object which contains:
  - resultxmlfile : path to the xml result file or directory of result files to be imported.
  - server : server which hosts the database (IP or URL).
  - user : user for database login.
  - password : password for database login.
  - database : database name.
  - recursive : if True, then the path is searched recursively for log files to be imported.
  - dryrun : if True, then just check the RQM authentication and show what would be done.
  - append : if True, then allow to append new result(s) to existing execution result UUID which is provided by -UUID argument.
  - UUID : UUID used to identify the import and version ID on TestResultWebApp.
  - config : configuration json file for component mapping information.

Returns:
(*no returns*)

4.13 Class: Logger

Imported by:

```
from PyTestLog2DB.pytestlog2db import Logger
```

Logger class for logging message.

4.13.1 Method: config

Configure Logger class.

Arguments:

- `output_console`
  / *Condition*: optional / *Type*: bool / *Default*: True /
  Write message to console output.
- `output_logfile`
  / *Condition*: optional / *Type*: str / *Default*: None /
  Path to log file output.
- `indent`
  / *Condition*: optional / *Type*: int / *Default*: 0 /
  Offset indent.
- `dryrun`
  / *Condition*: optional / *Type*: bool / *Default*: True /
  If set, a prefix as 'dryrun' is added for all messages.

Returns:

(*no returns*)

4.13.2 Method: log

Write log message to console/file output.

Arguments:

- `msg`
  / *Condition*: optional / *Type*: str / *Default*: "" /
  Message which is written to output.
- `color`
  / *Condition*: optional / *Type*: str / *Default*: None /
  Color style for the message.
- `indent`
  / *Condition*: optional / *Type*: int / *Default*: 0 /
  Offset indent.

Returns:

(*no returns*)

4.13.3 Method: log_warning

Write warning message to console/file output.

Arguments:

- `msg`
  / *Condition*: required / *Type*: str /
  Warning message which is written to output.

Returns:

(*no returns*)

4.13.4 Method: log_error

Write error message to console/file output.

Arguments:

- `msg`
  / *Condition*: required / *Type*: str /
  Error message which is written to output.
- `fatal_error`
  / *Condition*: optional / *Type*: bool / *Default*: False /
  If set, tool will terminate after logging error message.

Returns:

(*no returns*)

Chapter 5

Appendix

About this package:

Table 5.1: Package setup

Setup parameter	Value
Name	PyTestLog2DB
Version	0.1.1
Date	22.11.2022
Description	Imports pytest result(s) to TestResultWebApp database
Package URL	python-pytestlog2db
Author	Tran Duy Ngoan
Email	Ngoan.TranDuy@vn.bosch.com
Language	Programming Language :: Python :: 3
License	License :: OSI Approved :: Apache Software License
OS	Operating System :: OS Independent
Python required	>=3.0
Development status	Development Status :: 4 - Beta
Intended audience	Intended Audience :: Developers
Topic	Topic :: Software Development

Chapter 6

History

0.1.0	11/2022
Initial version	
0.1.1	22.11.2022
Initial implementation of PyTestLog2DB tool	