WeTest Documentation

Release 0.4.1-beta

Nicolas Senaud

Jul 04, 2017

CONTENTS

1 Project Documentation 1
1.1 WeTest 1
1.1.1 Installation 1
1.1.2 Run WeTest 2
1.1.3 Development 2
1.1.4 FAQ 3
2 Code Documentation 4
2.1 Modules: 4
2.1.1 wetest.command_line 4
2.1.2 wetest.testing.reader 6
2.1.3 wetest.testing.generator 8
2.1.4 wetest.report.generator 10
Python Module Index 13
Index 14

CHAPTER

ONE

PROJECT DOCUMENTATION

1.1 WeTest

1.1.1 Installation

Setup environment

You can use conda (miniconda) to setup a virtual environment. It is a good way to avoid poluting your system and to run Python 3 on CentOS 7 (note that currently WeTest is not compatible yet with Python 3).

```
wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh \
    && bash Miniconda2-latest-Linux-x86_64.sh
```

You should add the conda-forge repository to avoid a bug in readline version provided by conda:

```
conda config --add channels conda-forge # This step in only necessary the first␣↪time you use conda.
```

To create a new environment and source it, do:

```
conda create -n wetest python=2.7 source activate
wetest
```

After the first run, simply run source activate wetest to activate the environment.

Note: some dependencies are compatible yet with Python 3. I did not investigate further yet, but the middle-term goal is to be able to switch to it.

After usage, you can run:

```
source desactivate wetest
```

Install WeTest

```
python setup.py install
```

You can use wetest as a regular command line utility after that.

Install dependencies manually

This should not be necessary if you used setup.py to install wetest.

```
pip install -r requirements.txt
```

Setup pyepics

If you have an ESS EPICS Environment, the $PYEPICS_LIBCA environment variable should already been configured as follows. You can find full explanation here.

```
export PYEPICS_LIBCA=/opt/epics/bases/base-3.14.12.5/lib/centos7-x86_64/libca.so
```

1.1.2 Run WeTest

To run Wetest with a scenario file, use:

```
wetest --scenario <scenario.yaml>
```

Or:

```
wetest -s <scenario.yaml>
```

To run Wetest with a suite file (including one or several scenarios), use:

```
wetest --suite <suite.yaml>
```

Or:

```
wetest -S <suite.yaml>
```

1.1.3 Development

Using Vagrant

To ease the setup of the development environmente, I create a Vagrantfile (based on the ESS one, to have the EPICS setup). Since the native Virtualbox sharing folder is not reliable enough, the Vagrantfile is configured to use sshfs instead, which require an extension to be installed on the host:

```
vagrant plugin install vagrant-sshfs
```

You can then launch the VM provisionning, it will take several minutes to complete:

```
vagrant up --provider=virtualbox
```

When the setup is complete, you can begin to work as usual:

```
vagrant ssh cd /vagrant source
activate wetest
```

Please note that the file are synchronised between the host and the guest, so that you can work with your favorite editor on the host, and execute wetest on the guest.

Updating requirements.txt

If you modify dependencies, you can run:

```
pip freeze > requirements.txt
```

So that the new dependencies will be installed automatically for future users.

See here for requirements files documentation.

Running Sphinx document generator

```
cd doc/ make
html                    # To generate webpage

make latexpdf           # To generate PDF
```

Running unit tests

```
python setup.py test
```

To run them automatically whenever you save a Python file, you can use the following command (entr must be installed on your system):

```
find . -name "*.py" | entr python setup.py test
```

Measure test coverage

You can measure test coverage this way:

```
python -m coverage run --source=. setup.py test
```

And you can display the result with:

```
python -m coverage report -m
```

Measure code coverage

You can measure the code coverage with such a command:

```
python -m coverage run --source=. wetest/command_line.py \
                                                    -s tests/acceptance-demo.yaml \
                                                   -o /tmp/report.pdf
```

1.1.4 FAQ

I have the following error when I try to build the documentation: **make: sphinx-build: Command not found**

The installation path of Sphinx is not in your $PATH. Check where it is installed, on Linux with the --user flag it should be in ~/.local/bin.

CHAPTER

TWO

CODE DOCUMENTATION

2.1 Modules:

2.1.1 wetest.command_line

WeTest is the main module of the testing tool.

DummyClassForTesting class

class wetest.command_line.**DummyClassForTesting**(*methodName='runTest'*) A TestCase implementation where all tests will be skipped.

This is useful for testing: if you don't use this class but the regular one, unit testing of the code will probably fail, because the WeTest scenario probably will.

Runner class

class wetest.command_line.**Runner**(*suite*, *filename=None*) Run a test suite and save results in xUnit format.

run()

Run tests.

Returns filename the xUnit XML file name.

Returns results the unittests results.

new_suite_from function

wetest.command_line.**new_suite_from**(*scenario_file=None*, *suite_file=None*, *skip_all=False*) Create a test suite from a YAML file (suite or scenario).

Parameters

- **scenario_file** – A YAML scenario file path...
- **suite_file** – ... Or a YAML suite file path. At least one of them must be provided.
- **skip_all** – Skill all tests (should be useful for testing only).

Returns suite A test suite object.

Returns title Report title.

export_pdf function

wetest.command_line.**export_pdf**(*filename*, *tests*, *results*, *title*) Export tests results to PDF file. Parameters

- **filename** – The PDF filename.
- **tests** – The ran test case(s).
- **results** – The test result(s).
- **title** – The report's title.

run_tests function

wetest.command_line.**run_tests**(*scenario_file=None*, *suite_file=None*, *save_xml_to=None*) Execute scenario or test suite.

Parameters

- **scenario_file** – A YAML scenario file path...
- **suite_file** – ... Or a YAML suite file path. At least one of them must be provided.

Returns results Testing results.

Returns suite Test suite.

Returns title Report title.

main function

wetest.command_line.**main**()
Program's main entry point.

2.1.2 wetest.testing.reader

Read tests in YAML file.

Reader class

class wetest.testing.reader.**Reader**
A base class implementing a YAML file reader.

_version_is_supported(*major=None*, *minor=None*, *bugfix=None*) Check if configuration file format version is supported.

Parameters

- **major** – optionally force the major digit parameter (should be useful for testing only)
- **minor** – optionally force the minor digit parameter (should be useful for testing only)
- **bugfix** – optionally force the bugfix digit parameter (should be useful for testing only)

Returns True if version is supported, and raise exception UnsupportedFileFormat is any other cases.

get_deserialized()

Get the deserialized object.

Returns the deserialized object.

is_supported()
YAML file format version is supported.

Returns a boolean on whether it is supported or not.

is_valid()
YAML file format is valid.

Returns a boolean on whether it is valid or not.

SuiteReader class

class wetest.testing.reader.**Reader**
A base class implementing a YAML file reader.

_version_is_supported(*major=None*, *minor=None*, *bugfix=None*) Check if configuration file format version is supported.

Parameters

- **major** – optionally force the major digit parameter (should be useful for testing only)
- **minor** – optionally force the minor digit parameter (should be useful for testing only)
- **bugfix** – optionally force the bugfix digit parameter (should be useful for testing only)

Returns True if version is supported, and raise exception UnsupportedFileFormat is any other cases.

get_deserialized()
Get the deserialized object.

Returns the deserialized object.

is_supported()
YAML file format version is supported.

Returns a boolean on whether it is supported or not.

is_valid()
YAML file format is valid.

Returns a boolean on whether it is valid or not.

ScenarioReader class

class wetest.testing.reader.**Reader**
A base class implementing a YAML file reader.

_version_is_supported(*major=None*, *minor=None*, *bugfix=None*) Check if configuration file format version is supported.

Parameters

- **major** – optionally force the major digit parameter (should be useful for testing only)
- **minor** – optionally force the minor digit parameter (should be useful for testing only)
- **bugfix** – optionally force the bugfix digit parameter (should be useful for testing only)

Returns True if version is supported, and raise exception UnsupportedFileFormat is any other cases.

get_deserialized()
Get the deserialized object.

Returns the deserialized object.

is_supported()
YAML file format version is supported.

Returns a boolean on whether it is supported or not.

is_valid()
YAML file format is valid.

Returns a boolean on whether it is valid or not.

UnsupportedFileFormat class

class wetest.testing.reader.**UnsupportedFileFormat** Bad YAML file format exception.

Exception raised if YAML configuration file format version is newer than supported one.

2.1.3 wetest.testing.generator

Generates tests from YAML file.

TestData class

class wetest.testing.generator.**TestData**(*name=''*, *desc=''*, *getter=None*, *setter=None*, *get_value=None*, *set_value=None*, *prefix=''*, *delay=0*, *margin=0*) A generic test representation.

TestsDefaults class

class wetest.testing.generator.**TestsDefaults**(*delay=0*, *prefix=''*)
Tests default values.

TestsSequence class

class wetest.testing.generator.**TestsSequence**(*methodName='runTest'*) An empty class to be filled by test methods.

TestsSequence class will be dynamically fill of tests by TestsGenerator.

TestsGenerator class

class wetest.testing.generator.**TestsGenerator**(*tests_data*) TestGenerator generates unittest test cases from a YAML file.

_create_tests_list()
Create a list of TestData objects from deserialized file.

_get_prefix(*test_data*)

Assemble full prefix from prefix fields.

Parameters **test_data** – Test properties.

Returs Concatenate prefix(es).

_get_test_name(*index*, *subindex*) Compute test name.

The name will be composed of the following elements:

- *test* prefix: it is necessary for *unittest* to recognize the generated function as a test.
- zero-padded 3-digit index of the test: the test are executed in alphabetic order, so it is necessary to include there index to control there execution order.
- zero-padded 3-digit index of the sub-test.

Parameters

- **index** – The test index in the scenario.
- **subindex** – The test subindex for each (optional) steps of the test.

Returns Test's name.

_randomize_order()

Generate a random order for test execution.

Returns a list of integers from zero to tests count minus one, to access each test once in a random order from its index.

generates(*test_class*, *first_test_index=0*)

Generate *unittest* tests from configuration file.

Parameters

- **test_class** – Empty TestCase class to fill with tests.
- **first_test_index** – Index of the first step (0 by default). This is useful for test suite, when you want to begin to iterate from the last scenario index.

get_title()

Get test scenario title.

Returns Tests scenario title.

test_generator function

wetest.testing.generator.**test_generator**(*test_data*, *description*) Generates a *unnitest* test case from test's data.

Parameters

- **test_data** – a TestData instance (usually extracted from a YAML file).
- **description** – the test's docstring.

Returns a test case.

get_margin function

wetest.testing.generator.**get_margin**(*data*)

Get allowed margin between setter and getter values.

Parameters **data** – Test properties.

Returns allowed margin.

get_key function

wetest.testing.generator.**get_key**(*prefered*, *backup*, *key*) Get key by priority.

If the *key* exists in *prefered*, it value will be returned, or the *key* will be searched in *backup* instead. If it fails, returns None.

Parameters

- **prefered** – Prefered source.
- **backup** – Backup source.
- **key** – Key name.

Returns value found for key, or None.

add_doc function

wetest.testing.generator.**add_doc**(*value*)
Add docstring programatically to a function via a decorator.

Parameters **value** – Docstring value.

Returns Function's docstring.

2.1.4 wetest.report.generator

Create a PDF report from testing results.

ReportGenerator class

class wetest.report.generator.**ReportGenerator**(*test_suite*, *test_results*, *filename*, *title=''*) Generates a PDF report from test unit results.

save()
Save the report as a PDF file.

_TestInfo class

class wetest.report.generator.**_TestInfo**(*test_suite*, *test_results*) Combine information about tests success status.

_append_to_combined(*test*, *status*, *color*, *trace=None*) Append test information to combined test list.

Parameters

- **test** – The test function.
- **status** – A string describing the result status.
- **color** – The text color.

- **trace** – In case of failure, the stacktrace.

get_para_with_style function

wetest.report.generator.**get_para_with_style**(*text=''*, *bold=False*, *style='BodyText'*, *align='left'*, *color=None*) Get an initialized Paragraph object.

Parameters

- **text** – The text to be rendered.
- **bold** – A boolean on wether the text must be bold or not.
- **style** – The text style (from SampleStyleSheet object).
- **align** – The text alignment.

Returns An Initialized Paragraph.

PYTHON MODULE INDEX

W

wetest.command_line, 5 wetest.report.generator, 10
wetest.testing.generator, 8 wetest.testing.reader, 6

Python Module Index

INDEX

Symbols

_TestInfo (class in wetest.report.generator), 10
_append_to_combined() (wetest.report.generator._TestInfo method), 10
_create_tests_list() (wetest.testing.generator.TestsGeneratorReportGenerator method), 8
_get_prefix() (wetest.testing.generator.TestsGenerator method), 8
_get_test_name() (wetest.testing.generator.TestsGenerator method), 8
_randomize_order() (wetest.testing.generator.TestsGenerator method), 9
_version_is_supported() (wetest.testing.reader.Reader method), 6, 7

A

add_doc() (in module wetest.testing.generator), 10

D

DummyClassForTesting (class in wetest.command_line), 5

E

export_pdf() (in module wetest.command_line), 6

G

generates() (wetest.testing.generator.TestsGenerator method), 9
get_deserialized() (wetest.testing.reader.Reader method), 6, 7
get_key() (in module wetest.testing.generator), 10
get_margin() (in module wetest.testing.generator), 9

N

new_suite_from() (in module wetest.command_line), 5

R

Reader (class in wetest.testing.reader), 6, 7
ReportGenerator (class in wetest.report.generator), 10
run() (wetest.command_line.Runner method), 5
run_tests() (in module wetest.command_line), 6
Runner (class in wetest.command_line), 5

S

save() (wetest.report.generator.ReportGenerator method), 10

T

test_generator() (in module wetest.testing.generator), 9
TestData (class in wetest.testing.generator), 8
TestsDefaults (class in wetest.testing.generator), 8
TestsGenerator (class in wetest.testing.generator), 8
TestsSequence (class in wetest.testing.generator), 8

U

get_para_with_style() (in module wetest.report.generator), 11
get_title() (wetest.testing.generator.TestsGenerator method), 9

I

is_supported() (wetest.testing.reader.Reader method), 7, 8
is_valid() (wetest.testing.reader.Reader method), 7, 8

M

main() (in module wetest.command_line), 6

UnsupportedFileFormat (class in wetest.testing.reader), 8

W

wetest.command_line (module), 5
wetest.report.generator (module), 10
wetest.testing.generator (module), 8
wetest.testing.reader (module), 6