Virtue

Release 2023.7.6.dev8+g5f7e849

Julian Berman

Aug 22, 2023

CONTENTS

1 Usage **3**

2 Contributing **5**

3 Contents **7**

3.1 API Reference 7

Python Module Index **13**

Index **15**

`virtue` is a modern, extensible, unittest compliant test runner.

It is *not* a test framework (it doesn't contain a `TestCase` subclass and it never will).

CHAPTER

ONE

USAGE

Running a `unittest`-based suite works essentially as it does for `twisted`'s `trial`, i.e.:

```
$ python -m virtue mypackage.tests
```

will run the tests subpackage of a given importable package.

More docs are coming. Sorry.

CHAPTER

TWO

CONTRIBUTING

I'm Julian Berman.

`virtue` is on GitHub.

Get in touch, via GitHub or otherwise, if you've got something to contribute, it'd be most welcome!

You can also generally find me on Libera (nick: `Julian`) in various channels, including `#python`.

If you feel overwhelmingly grateful, you can also sponsor me.

CHAPTER

THREE

CONTENTS

3.1 API Reference

The virtuous test runner.

3.1.1 virtue.loaders

Loaders take a named test case and load the test appropriately.

class virtue.loaders.**AttributeLoader**(*cls: type*, *attribute: str*)

I load a test case by instantiating a class with a given attribute name.

This is the typical way that `unittest.TestCase` methods are loaded: by calling `TestCase("test_something")` (and then by calling `run()` on the resulting instance to run the selected test method).

cls: *type*

attribute: *str*

load()

Load as a single test.

class virtue.loaders.**ModuleLoader**(*locator: ObjectLocator*, *module: twisted.python.modules.PythonModule*)

I load a test case by locating tests in the module with the given name.

locator: *ObjectLocator*

module: *twisted.python.modules.PythonModule*

load()

Load all test cases in the module.

3.1.2 virtue.locators

Loaders find tests which are referenced by names, preparing them for running.

exception virtue.locators.**UnableToLoad**

A test couldn't be loaded.

virtue.locators.**prefixed_by**(*prefix*)

Make a callable returning True for names starting with the given prefix.

The returned callable takes two arguments, the attribute or name of the object, and possibly its corresponding value (which is ignored), as suitable for use with *ObjectLocator.is_test_module* and *ObjectLocator.is_test_method*.

virtue.locators.**inherits_from_TestCase**(*attr*, *cls*)

Return true if a class inherits from unittest.TestCase.

class virtue.locators.**ObjectLocator**(*is_test_method=<function prefixed_by.<locals>.prefixed_by_>*, *is_test_class=<function inherits_from_TestCase>*, *is_test_module=<function prefixed_by.<locals>.prefixed_by_>*)

I locate test cases on an object: a package, module or test class.

Parameters

- **is_test_method** (*collections.abc.Callable*) – decide whether the provided object is a test method or not. By default, callable objects whose names (`__name__``s) start with ``test_` are considered test methods.
- **is_test_class** (*collections.abc.Callable*) – decide whether the provided object is a test class or not. By default, objects inheriting from unittest.TestCase are considered test cases.
- **is_test_module** (*collections.abc.Callable*) – decide whether the provided object is a test module or not. By default, modules whose names start with `test_` are considered to be test modules.

is_test_method

Whether an object is a test method or not

is_test_class

Whether an object is a test class or not

is_test_module

Whether an object is a test module or not

locate_by_name(*name*)

Locate any tests found in the object referred to by the given name.

The name should be a fully qualified object name. (E.g., the fully qualified object name of this function is `virtue.locators.ObjectLocator.locate_by_name`).

A path may also alternatively used, but no PYTHONPATH modification will be done, so the file must be importable without modification.

locate_in(*obj*)

Attempt to locate the test cases in the given object (of any kind).

locate_in_package(*package*)

Locate all of the test cases contained in the given package.

locate_in_module(*module*)

Locate all of the test cases contained in the given module.

locate_in_class(*cls*)

Locate the methods on the given class that are test cases.

3.1.3 virtue.reporters

Outputting and reporting for virtuous test runs.

class virtue.reporters.**Outputter**(*colored=True*, *indent=' '*, *line_width=120*)

An outputter converts test results to renderable strings.

FAILED = 'FAILED'

PASSED = 'PASSED'

ERROR = '[ERROR]'

FAIL = '[FAIL]'

OK = '[OK]'

SKIPPED = '[SKIPPED]'

EXPECTED_FAILURE = '[XFAIL]'

UNEXPECTED_SUCCESS = '[UNEXPECTED SUCCESS]'

run_started()

Output nothing.

run_stopped(*recorder*, *runtime*)

Output all the messages stored for later, as well as a final summary.

test_started(*test*)

Output the test name.

test_stopped(*test*)

Output nothing.

test_errored(*test*, *exc_info*)

Output an error.

test_failed(*test*, *exc_info*)

Output a failure.

test_skipped(*test*, *reason*)

Output a skip.

test_expectedly_failed(*test*, *exc_info*)

Output an expected failure.

test_unexpectedly_succeeded(*test*)

Output an unexpected success.

test_succeeded(*test*)

Output a success.

subtest_succeeded(*test*, *subtest*)

Output nothing.

subtest_failed(*test*, *subtest*, *exc_info*)

Output a failed subtest.

subtest_errored(*test*, *subtest*, *exc_info*)

Output an errored subtest.

class virtue.reporters.**Counter**(*errors: int = 0, failures: int = 0, expected_failures: int = 0, unexpected_successes: int = 0, successes: int = 0, subtest_successes: int = 0, subtest_failures: int = 0, subtest_errors: int = 0*)

A counter is a recorder that does not hold references to tests it sees.

errors: int

failures: int

expected_failures: int

unexpected_successes: int

successes: int

subtest_successes: int

subtest_failures: int

subtest_errors: int

shouldStop = False

property count

Return a total count of all tests.

property testsRun

Return a total count of all tests.

startTest(*test*)

stopTest(*test*)

addError(*test*, *exc_info*)

addFailure(*test*, *exc_info*)

addExpectedFailure(**args*, ***kwargs*)

addUnexpectedSuccess(*test*)

addSuccess(*test*)

addDuration(*test*, *elapsed*)

addSubTest(*test*, *subtest*, *outcome*)

class virtue.reporters.**Recorder**(*errors: PVector = pvector([]), failures: PVector = pvector([]), skips: PVector = pvector([]), successes: PVector = pvector([]), expected_failures: PVector = pvector([]), unexpected_successes: PVector = pvector([]), subtest_successes: PMap = pmap({}), subtest_failures: PMap = pmap({}), subtest_errors: PMap = pmap({})*)

Record test results for later inspection.

errors: PVector

failures: PVector

skips: PVector

successes: PVector

expected_failures: PVector

unexpected_successes: PVector

subtest_successes: PMap

subtest_failures: PMap

subtest_errors: PMap

shouldStop = False

property testsRun

property subtests

startTestRun()

stopTestRun()

startTest(*test*)

stopTest(*test*)

addError(*test*, *exc_info*)

addFailure(*test*, *exc_info*)

addExpectedFailure(*test*, *exc_info*)

addSkip(*test*, *reason*)

addUnexpectedSuccess(*test*)

addSuccess(*test*)

addDuration(*test*, *elapsed*)

addSubTest(*test*, *subtest*, *outcome*)

wasSuccessful()

class virtue.reporters.**ComponentizedReporter**(*outputter: ~virtue.reporters.Outputter = _Nothing.NOTHING, recorder=_Nothing.NOTHING, stream=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>, time=<built-in function time>*)

Combine together outputting and recording capabilities.

outputter: *Outputter*

failfast = False

shouldStop = False

property testsRun

startTestRun()

stopTestRun()

startTest(*test*)

stopTest(*test*)

addError(*test*, *exc_info*)

addFailure(*test*, *exc_info*)

addSkip(*test*, *reason*)

addExpectedFailure(*test*, *exc_info*)

addUnexpectedSuccess(*test*)

addSuccess(*test*)

addDuration(*test*, *elapsed*)

addSubTest(*test*, *subtest*, *outcome*)

wasSuccessful()

3.1.4 virtue.runner

Runners execute loaded tests.

virtue.runner.**run**(*tests=()*, *reporter=None*, *stop_after=None*)

Run the tests that are loaded by each of the strings provided.

Parameters

- **tests** (*collections.abc.Iterable*) – the collection of tests (specified as `str` s) to run
- **reporter** (*twisted.trial.itrial.IReporter*) – a reporter to use for the run. If unprovided, the default is to return a *virtue.reporters.Counter* (which produces no output).
- **stop_after** (*int*) – a number of non-successful tests to allow before stopping the run.

PYTHON MODULE INDEX

V

virtue, 7
virtue.loaders, 7
virtue.locators, 8
virtue.reporters, 9
virtue.runner, 12

INDEX

A

addDuration() (*virtue.reporters.ComponentizedReporter method*), 12
addDuration() (*virtue.reporters.Counter method*), 10
addDuration() (*virtue.reporters.Recorder method*), 11
addError() (*virtue.reporters.ComponentizedReporter method*), 12
addError() (*virtue.reporters.Counter method*), 10
addError() (*virtue.reporters.Recorder method*), 11
addExpectedFailure() (*virtue.reporters.ComponentizedReporter method*), 12
addExpectedFailure() (*virtue.reporters.Counter method*), 10
addExpectedFailure() (*virtue.reporters.Recorder method*), 11
addFailure() (*virtue.reporters.ComponentizedReporter method*), 12
addFailure() (*virtue.reporters.Counter method*), 10
addFailure() (*virtue.reporters.Recorder method*), 11
addSkip() (*virtue.reporters.ComponentizedReporter method*), 12
addSkip() (*virtue.reporters.Recorder method*), 11
addSubTest() (*virtue.reporters.ComponentizedReporter method*), 12
addSubTest() (*virtue.reporters.Counter method*), 10
addSubTest() (*virtue.reporters.Recorder method*), 11
addSuccess() (*virtue.reporters.ComponentizedReporter method*), 12
addSuccess() (*virtue.reporters.Counter method*), 10
addSuccess() (*virtue.reporters.Recorder method*), 11
addUnexpectedSuccess() (*virtue.reporters.ComponentizedReporter method*), 12
addUnexpectedSuccess() (*virtue.reporters.Counter method*), 10
addUnexpectedSuccess() (*virtue.reporters.Recorder method*), 11
attribute (*virtue.loaders.AttributeLoader attribute*), 7
AttributeLoader (*class in virtue.loaders*), 7

C

cls (*virtue.loaders.AttributeLoader attribute*), 7
ComponentizedReporter (*class in virtue.reporters*), 11
count (*virtue.reporters.Counter property*), 10
Counter (*class in virtue.reporters*), 10

E

ERROR (*virtue.reporters.Outputter attribute*), 9
errors (*virtue.reporters.Counter attribute*), 10
errors (*virtue.reporters.Recorder attribute*), 11
EXPECTED_FAILURE (*virtue.reporters.Outputter attribute*), 9
expected_failures (*virtue.reporters.Counter attribute*), 10
expected_failures (*virtue.reporters.Recorder attribute*), 11

F

FAIL (*virtue.reporters.Outputter attribute*), 9
FAILED (*virtue.reporters.Outputter attribute*), 9
failfast (*virtue.reporters.ComponentizedReporter attribute*), 12
failures (*virtue.reporters.Counter attribute*), 10
failures (*virtue.reporters.Recorder attribute*), 11

I

inherits_from_TestCase() (*in module virtue.locators*), 8
is_test_class (*virtue.locators.ObjectLocator attribute*), 8
is_test_method (*virtue.locators.ObjectLocator attribute*), 8
is_test_module (*virtue.locators.ObjectLocator attribute*), 8

L

load() (*virtue.loaders.AttributeLoader method*), 7
load() (*virtue.loaders.ModuleLoader method*), 7
locate_by_name() (*virtue.locators.ObjectLocator method*), 8
locate_in() (*virtue.locators.ObjectLocator method*), 8

locate_in_class() (*virtue.locators.ObjectLocator method*), 9
locate_in_module() (*virtue.locators.ObjectLocator method*), 8
locate_in_package() (*virtue.locators.ObjectLocator method*), 8
locator (*virtue.loaders.ModuleLoader attribute*), 7

M

module
    virtue, 7
    virtue.loaders, 7
    virtue.locators, 8
    virtue.reporters, 9
    virtue.runner, 12
module (*virtue.loaders.ModuleLoader attribute*), 7
ModuleLoader (*class in virtue.loaders*), 7

O

ObjectLocator (*class in virtue.locators*), 8
OK (*virtue.reporters.Outputter attribute*), 9
Outputter (*class in virtue.reporters*), 9
outputter (*virtue.reporters.ComponentizedReporter attribute*), 12

P

PASSED (*virtue.reporters.Outputter attribute*), 9
prefixed_by() (*in module virtue.locators*), 8

R

Recorder (*class in virtue.reporters*), 10
run() (*in module virtue.runner*), 12
run_started() (*virtue.reporters.Outputter method*), 9
run_stopped() (*virtue.reporters.Outputter method*), 9

S

shouldStop (*virtue.reporters.ComponentizedReporter attribute*), 12
shouldStop (*virtue.reporters.Counter attribute*), 10
shouldStop (*virtue.reporters.Recorder attribute*), 11
SKIPPED (*virtue.reporters.Outputter attribute*), 9
skips (*virtue.reporters.Recorder attribute*), 11
startTest() (*virtue.reporters.ComponentizedReporter method*), 12
startTest() (*virtue.reporters.Counter method*), 10
startTest() (*virtue.reporters.Recorder method*), 11
startTestRun() (*virtue.reporters.ComponentizedReporter method*), 12
startTestRun() (*virtue.reporters.Recorder method*), 11
stopTest() (*virtue.reporters.ComponentizedReporter method*), 12
stopTest() (*virtue.reporters.Counter method*), 10
stopTest() (*virtue.reporters.Recorder method*), 11
stopTestRun() (*virtue.reporters.ComponentizedReporter method*), 12
stopTestRun() (*virtue.reporters.Recorder method*), 11
subtest_errored() (*virtue.reporters.Outputter method*), 10
subtest_errors (*virtue.reporters.Counter attribute*), 10
subtest_errors (*virtue.reporters.Recorder attribute*), 11
subtest_failed() (*virtue.reporters.Outputter method*), 10
subtest_failures (*virtue.reporters.Counter attribute*), 10
subtest_failures (*virtue.reporters.Recorder attribute*), 11
subtest_succeeded() (*virtue.reporters.Outputter method*), 9
subtest_successes (*virtue.reporters.Counter attribute*), 10
subtest_successes (*virtue.reporters.Recorder attribute*), 11
subtests (*virtue.reporters.Recorder property*), 11
successes (*virtue.reporters.Counter attribute*), 10
successes (*virtue.reporters.Recorder attribute*), 11

T

test_errored() (*virtue.reporters.Outputter method*), 9
test_expectedly_failed() (*virtue.reporters.Outputter method*), 9
test_failed() (*virtue.reporters.Outputter method*), 9
test_skipped() (*virtue.reporters.Outputter method*), 9
test_started() (*virtue.reporters.Outputter method*), 9
test_stopped() (*virtue.reporters.Outputter method*), 9
test_succeeded() (*virtue.reporters.Outputter method*), 9
test_unexpectedly_succeeded() (*virtue.reporters.Outputter method*), 9
testsRun (*virtue.reporters.ComponentizedReporter property*), 12
testsRun (*virtue.reporters.Counter property*), 10
testsRun (*virtue.reporters.Recorder property*), 11

U

UnableToLoad, 8
UNEXPECTED_SUCCESS (*virtue.reporters.Outputter attribute*), 9
unexpected_successes (*virtue.reporters.Counter attribute*), 10
unexpected_successes (*virtue.reporters.Recorder attribute*), 11

V

virtue
    module, 7

virtue.loaders
    module, 7
virtue.locators
    module, 8
virtue.reporters
    module, 9
virtue.runner
    module, 12

W

wasSuccessful() (*virtue.reporters.ComponentizedReporter method*), 12
wasSuccessful() (*virtue.reporters.Recorder method*), 11