pytest-dependency Documentation

Release 0.2

Rolf Krahl

Dec 26, 2017

Contents

1 Content of the documentation **3**
- 1.1 About pytest-dependency . . . 3
- 1.2 Installation instructions . . . 4
- 1.3 Using pytest-dependency . . . 5
- 1.4 Reference . . . 8

Python Module Index **9**

This pytest plugin manages dependencies of tests. It allows to mark some tests as dependent from other tests. These tests will then be skipped if any of the dependencies did fail or has been skipped.

CHAPTER 1

Content of the documentation

1.1 About pytest-dependency

This module is a plugin for the popular Python testing framework pytest. It manages dependencies of tests: you may mark some tests as dependent from other tests. These tests will then be skipped if any of the dependencies did fail or has been skipped.

1.1.1 What is the purpose?

In the theory of good test design, tests should be self-contained and independent. Each test should cover one single issue, either verify that one single feature is working or that one single bug is fixed. Tests should be designed to work in any order independent of each other.

So far the theory. The practice is often more complicated then that. Sometimes, the principle of independency of tests is simply unrealistic or impractical. Program features often depend on each other. If some feature B depends on another feature A in such a way that B cannot work without A, then it may simply be pointless to run the test for B unless the test for A has succeeded. Another case may be if the subject of the tests has an internal state that unavoidably is influenced by the tests. In this situation it may happen that test A, as a side effect, sets the system in some state that is the precondition to be able to run test B. Again, in this case it would be pointless to try running test B unless test A has been run successful.

It should be emphasized however that the principle of independency of tests is still valid. Before using pytest-dependency, it is still advisable to reconsider your test design and to avoid dependencies of tests whenever possible, rather then to manage these dependencies.

1.1.2 How does it work?

The pytest-dependency module defines a marker that can be applied to tests. The marker accepts an argument that allows to list the dependencies of the test. Both tests, the dependency and the dependent test should be decorated with the marker. Behind the scenes, the marker arranges for the result of the test to be recorded internally. If a list of dependencies has been given as argument, the marker verifies that a successful outcome of all the dependencies has been registered previously and causes a skip of the test if this was not the case.

1.1.3 Why is this useful?

The benefit of skipping dependent tests is the same as for skipping tests in general: it avoids cluttering the test report with useless and misleading failure reports from tests that have been known beforehand not to work in this particular case.

If tests depend on each other in such a way that test B cannot work unless test A has been run successfully, a failure of test A will likely result in failure messages from both tests. But the failure message from test B will not be helpful in any way. It will only distract the user from the real issue that is the failure of test A. Skipping test B in this case will help the user to concentrate on those results that really matter.

1.1.4 Copyright and License

1.2 Installation instructions

1.2.1 System requirements

- Python 2.6, 2.7, or 3.1 and newer. Python 2.6 requires patching the sources, see below.
- setuptools.
- pytest 2.8.0 or newer.

1.2.2 Download

The latest release version of pytest-dependency source can be found at PyPI, see

https://pypi.python.org/pypi/pytest_dependency

1.2.3 Installation

1. Download the sources, unpack, and change into the source directory.
2. Build (optional):

```
$ python setup.py build
```

3. Test (optional):

```
$ python -m pytest
```

4. Install:

```
$ python setup.py install
```

The last step might require admin privileges in order to write into the site-packages directory of your Python installation.

If you are using Python 2.6, apply python2_6.patch after the first step:

1a. Patch:

```
$ patch -p1 < python2_6.patch
```

It removes the use of certain language features (dict comprehensions) that were introduced in Python 2.7.

For production use, it is always recommended to use the latest release version from PyPI, see above. If you build from the development sources that can be found at GitHub, please note that python2_6.patch is generated dynamically and not in the source repository.

1.3 Using pytest-dependency

The plugin defines a new marker *pytest.mark.dependency()*.

1.3.1 Basic usage

Consider the following example test module:

```
import pytest

@pytest.mark.dependency()
@pytest.mark.xfail(reason="deliberate fail")
def test_a():
    assert False

@pytest.mark.dependency()
def test_b():
    pass

@pytest.mark.dependency(depends=["test_a"])
def test_c():
    pass

@pytest.mark.dependency(depends=["test_b"])
def test_d():
    pass

@pytest.mark.dependency(depends=["test_b", "test_c"])
def test_e():
    pass
```

All the tests are decorated with *pytest.mark.dependency()*. This will cause the test results to be registered internally and thus other tests may depend on them. The list of dependencies of a test may be set in the optional *depends* argument to the marker. The first test has deliberately been set to fail to illustrate the effect. We will get the following resuts:

test_a deliberatly fails.

test_b succeeds.

test_c will be skipped because it depends on *test_a*.

test_d depends on *test_b* which did succeed. It will be run and succeed as well.

test_e depends on *test_b* and *test_c*. *test_b* did succeed, but *test_c* has been skipped. So this one will also be skipped.

1.3.2 Naming tests

Tests are referenced by their name in the *depends* argument. The default for this name is the node ID defined by pytest, that is the name of the test function, extended by the parameters if applicable. As these node IDs may become complicated, the name can be overridden by an explicit *name* argument to the marker. The following example works exactly as the last one, only the test names are explicitely set:

```
import pytest

@pytest.mark.dependency(name="a")
@pytest.mark.xfail(reason="deliberate fail")
def test_a():
    assert False

@pytest.mark.dependency(name="b")
def test_b():
    pass

@pytest.mark.dependency(name="c", depends=["a"])
def test_c():
    pass

@pytest.mark.dependency(name="d", depends=["b"])
def test_d():
    pass

@pytest.mark.dependency(name="e", depends=["b", "c"])
def test_e():
    pass
```

1.3.3 Parametrized tests

In the same way as the `pytest.mark.skip()` and `pytest.mark.xfail()` markers, the *`pytest.mark.dependency()`* marker may be applied to individual test instances in the case of parametrized tests. Consider the following example:

```
import pytest

@pytest.mark.parametrize("x,y", [
    pytest.mark.dependency(name="a1")((0,0)),
    pytest.mark.dependency(name="a2")(pytest.mark.xfail((0,1))),
    pytest.mark.dependency(name="a3")((1,0)),
    pytest.mark.dependency(name="a4")((1,1))
])
def test_a(x,y):
    assert y <= x

@pytest.mark.parametrize("u,v", [
```

```
    pytest.mark.dependency(name="b1", depends=["a1", "a2"])((1,2)),
    pytest.mark.dependency(name="b2", depends=["a1", "a3"])((1,3)),
    pytest.mark.dependency(name="b3", depends=["a1", "a4"])((1,4)),
    pytest.mark.dependency(name="b4", depends=["a2", "a3"])((2,3)),
    pytest.mark.dependency(name="b5", depends=["a2", "a4"])((2,4)),
    pytest.mark.dependency(name="b6", depends=["a3", "a4"])((3,4))
])
def test_b(u,v):
    pass

@pytest.mark.parametrize("w", [
    pytest.mark.dependency(name="c1", depends=["b1", "b2", "b6"])(1),
    pytest.mark.dependency(name="c2", depends=["b2", "b3", "b6"])(2),
    pytest.mark.dependency(name="c3", depends=["b2", "b4", "b6"])(3)
])
def test_c(w):
    pass
```

The test instance *test_a[0-1]*, named *a2* in the *pytest.mark.dependency()* marker, is going to fail. As a result, the dependent tests *b1*, *b4*, *b5*, and in turn *c1* and *c3* will be skipped.

1.3.4 Marking dependencies at runtime

Sometimes, dependencies of test instances are too complicated to be formulated explicitely beforehand using the *pytest.mark.dependency()* marker. It may be easier to compile the list of dependencies of a test at run time. In such cases, the function `pytest_dependency.depends()` comes handy. Consider the following example:

```
import pytest
from pytest_dependency import depends

@pytest.mark.dependency()
def test_a():
    pass

@pytest.mark.dependency()
@pytest.mark.xfail(reason="deliberate fail")
def test_b():
    assert False

@pytest.mark.dependency()
def test_c(request):
    depends(request, ["test_b"])
    pass

@pytest.mark.dependency()
def test_d(request):
    depends(request, ["test_a", "test_c"])
    pass
```

Tests *test_c* and *test_d* set their dependencies at runtime calling `pytest_dependency.depends()`. The first argument is the values of the *request* pytest fixture, the second argument is the list of dependencies. It has the same effect as passing this list as the *depends* argument to the *pytest.mark.dependency()* marker.

1.4 Reference

`@pytest.mark.`**`dependency`**(*name=None*, *depends=[]*)

Mark a test to be used as a dependency for other tests or to depend on other tests.

This will cause the test results to be registered internally and thus other tests may depend on the test. The list of dependencies for the test may be set in the depends argument.

Parameters

- **`name`** (`str`) – the name of the test to be used for referencing by dependent tests. If not set, it defaults to the node ID defined by pytest, that is the name of the test function, extended by the parameters if applicable.
- **`depends`** (iterable of `str`) – dependencies, a list of names of tests that this test depends on. The test will be skipped unless all of the dependencies have been run successfully. The dependencies must also have been decorated by the marker.

Python Module Index

p

pytest_dependency, 8

Index

P

pytest.mark.dependency() (built-in function), 8
pytest_dependency (module), 8