Pytest: Sharing fixtures across tests using class level scopes

On Mar 16, 2017 by Mitesh Shah

I've been using pytest for writing tests because I absolutely love the simple assert systems. Although I'd love to write a small article detailing an easy introduction to pytest, I'd be digressing from the original title.

The Problem

What exactly is the problem I'll be describing: using pytest to share the same instance of setup and teardown code among multiple tests.

So instead of

```
setup --> test_1 --> teardown
setup --> test_2 --> teardown
```

We want:

```
setup --> test_1 --> test_2 --> teardown
```

Why?

While testing, there is a need to create an environment for the test like a database, or an empty directory.

Example:

```
# I'm taking an example of a flask test application
# In pytest we do this using advanced fixtures
@pytest.fixtures(scope='function')
def client():
    # create a database
    # at the start of the test
    db.create_all()

    app_client = app.test_client()
    app_client.db = db
    yield app_client
```

```
    # remove the database
    # at the end of the test
    db.drop_all()
```

Most of the times we can make-do with a completely fresh environment for every test function we write, but at some point you'll come across for a need to group multiple tests with the same instance of an environment.

Example, let's say we want the user to register, login, and check his details all within the same database instance:

```
def test_user_register(client):
    ...

def test_user_login(client):
    ...

def test_user_details(client):
    ...
```

Pytest provides "scopes" for such groupings. Here pytest will run the `client` fixture for every function separately, since it's scoped as a function. That means our database will lose the changes made by the register function when it runs the

login function. To avoid losing these changes, we effectively want to *share* the same fixture instance across these tests.

The Solution

There is an easy way to solve this, we can just extract all three tests into it's own module and create a *module* scoped fixture using `@pytest.fixtures(scope='module')`, everything should work as expected. But what if we need to do this across several tests which belong to the several user stories?

Pytest documentation mostly highlights three scope levels namely: *function, module, and session* scopes, but there is another scope available, which is the *class* scope.

In a class level scope, we directly inject our environment into the class as instance variables, which are then shared among all the methods of the class.

First we create a class scoped fixture:

```
@pytest.fixture(scope='class')
def class_client(request):
    # setup code
    db = ...
    db.create_all()
```

```
    # inject class variables
    request.cls.db = db
    yield

    # teardown code
    db.drop_all()
```

Here, as you can see we're adding the database object, which needs to be shared among all the class methods, into the class of the current function which is run.

```
request.cls.db = db
```

This is done using the `request` parameter provided by pytest to give the request of the current test function.

To use this fixture on a class, we simply deocarate the class as follows:

```
@pytest.mark.usefixtures('class_client')
class TestUserStory:
    def test_user_register(self):
        self.db.add(...)
```

```
    def test_user_login(self):
        # changes made in register are persisted
        # in the database object
        self.db.query(...)

    def test_user_details(self):
        ...
```

Now every function in the class can automatically access the injected variables from the fixture as `self.variable`.

That's about it. I hope the explanation was clear and concise, any sort of feedback is appreciated! Thanks.

« Custom views using django-adminplus, a quick tutorial

Fix Django Invalid HTTP_HOST header emails »

Dhiraj Dwarapudi • a year ago

This is a nice feature. Thanks for sharing this.

^ | v • Reply • Share ›

Drafter250 • a year ago

Im still confused how did the db variable inside the function get added to the class? are you saying every variable you create inside the function gets added to the test class? or is the test class being passed in to the fixture function as the "request" argument?

^ | v • Reply • Share ›

Kapil Mathur • a year ago • edited

Hi Mitesh... Thats an amazing article and it was really very helpful. I have one important question though.. what if i want to define this class-scope fixture in somewhere else.. and if i can do that.. will that class-scope fixture still be available to all my classes where i use it.. ??

^ | v • Reply • Share ›

oxalorg rogue ➚ Kapil Mathur • a year ago

Hey Kapil. Yes Fixtures can be shared in different files, for that you need to allow pytest to discover them automatically. This can be done by importing / declaring those fixtures in the `conftest.py` file.

You can read more about it here in the docs: https://docs.pytest.org/en/...

^ | v • Reply • Share ›

}

rss