pytest фикстуры / pytest fixtures

Литус Ярослав / Yaroslav Litus

Группа тестирования поиска / search testing group
Поисковый Портал Спутник / sputnik.ru

Поговорим о...

1. Понятие фикстуры
2. Фикстуры в pytest
   - Основы
   - Кэширование
   - Зависимости между фикстурами
   - skip/fail в фикстурах
   - Использование маркеров в фикстурах
   - Autouse-фикстуры
3. Всё вместе
4. Демонстрация
5. Полезные ресурсы

Понятие фикстуры

- In software testing, a test fixture is a fixed state of the software under test used as a baseline for running tests; also known as the test context.

Понятие фикстуры

- In software testing, a test fixture is a fixed state of the software under test used as a baseline for running tests; also known as the test context.
- В классических xUnit-фреймворках фикстуры реализуются через описание действий, относящихся к Setup и к Teardown. Например в unittest необходимо переопределить методы setUp() и к tearDown() класса TestCase.

Понятие фикстуры

- In software testing, a test fixture is a fixed state of the software under test used as a baseline for running tests; also known as the test context.
- В классических xUnit-фреймворках фикстуры реализуются через описание действий, относящихся к Setup и к Teardown. Например в unittest необходимо переопределить методы setUp() и к tearDown() класса TestCase.
- pytest вводит понятие функции-фикстуры, которая передается в тестовую функцию/метод в виде funcarg-а (позиционного аргумента).

Понятие фикстуры

- In software testing, a test fixture is a fixed state of the software under test used as a baseline for running tests; also known as the test context.
- В классических xUnit-фреймворках фикстуры реализуются через описание действий, относящихся к Setup и к Teardown. Например в unittest необходимо переопределить методы setUp() и к tearDown() класса TestCase.
- pytest вводит понятие функции-фикстуры, которая передается в тестовую функцию/метод в виде funcarg-а (позиционного аргумента).
- Фикстуры в pytest реализуют паттерн Dependency Injection.

```python
import pytest

@pytest.fixture
def foo():
    return 42

def test_foo(foo):
    assert foo == 42

class TestBar:
    @pytest.fixture
    def bar(self, request):
        def fin():
            print('Teardown of fixture bar')
        request.addfinalizer(fin)
        return 7

    def test_bar(self, foo, bar):
        assert foo != bar
```

yield_fixture

```
1  import pytest
2
3  @pytest.fixture
4  def foo():
5      return 42
6
7  def test_foo(foo):
8      assert foo == 42
9
10 class TestBar:
11     @pytest.yield_fixture
12     def bar(self):
13         yield 7
14         print('Teardown of fixture bar')
15
16     def test_bar(self, foo, bar):
17         assert foo != bar
```

У декораторов `fixture` и `yield_fixture` есть аргумент **scope**:

```
1  import pytest
2  @pytest.yield_fixture(scope='session')
3  def foo():
4      print('session setup')
5      yield 'foo'
6      print('session finalizer')
7
8  @pytest.yield_fixture(scope='function')  # default scope
9  def bar():
10     print('funtion setup')
11     yield 'bar'
12     print('function finalizer')
13
14 def test_one(foo, bar):
15     pass
16
17 def test_two(foo, bar):
18     pass
```

```
$ py.test-3 fixtures-caching.py -s -v
================================== test session starts ==================
platform linux -- Python 3.5.2, pytest-2.8.7, py-1.4.31, pluggy-0.3.1 -
collected 2 items

fixtures-caching.py::test_one session setup
funtion setup
PASSEDfunction finalizer

fixtures-caching.py::test_two funtion setup
PASSEDfunction finalizer
session finalizer
============================== 2 passed in 0.00 seconds ================
```

Доступные значения scope:

`function`, `class`, `module`, `session`

Зависимости между фикстурами

Фикстуры могут использовать другие фикстуры:

```
1  @pytest.fixture(scope='session')
2  def db_conn():
3      return create_db_conn()
4
5  @pytest.yield_fixture(scope='module')
6  def db_table(db_conn):
7      table = create_table(db_conn, 'foo')
8      yield table
9      drop_table(db_conn, table)
10
11 def test_bar(db_table):
12     pass
```

skip/fail в фикстурах

Вызов `pytest.skip()` или `pytest.fail()` внутри тестовой функции явно пропускает или фэйлит тест.

То же самое возможно использовать в фикстурах:

```
1  @pytest.fixture(scope='session')
2  def redis_client():
3      servers = ['localhost', 'venera.clockhouse']
4      for hostname in servers:
5          try:
6              return redis.StrictRedis(hostname)
7          except redis.ConnectionError:
8              continue
9      else:
10         pytest.skip('No Redis server found')
```

Таким образом все тесты, зависящие от такой фикстуры будут пропущены (skip) или сломаны (fail).

Использование маркеров в фикстурах

Используя доступ из фикстуры к маркерам зависимой функции, возможно реализовать настраиваемую фикстуру:

```
1  @pytest.fixture
2  def mongo_client(request):
3      marker = request.node.get_marker('mongo_db')
4      if not marker:
5          db = 'TestDB'
6      else:
7          db = marker.args[0]
8          # a better (and reliable) way to do this:
9          # db = (lambda x: x)(*marker.args, **marker.kwargs)
10     return pymongo.MongoClient('127.0.0.1/{}'.format(db))
11
12 @pytest.mark.mongo_db('Users')
13 def test_something(mongo_client):
14     pass
```

Autouse-фикстуры

```
1  @pytest.mark.linux
2  def test_mem_stack():
3      assert MemSizes().stack == 42
4
5  @pytest.fixture(autouse=True)
6  def _platform_skip(request):
7      marker = request.node.get_marker('linux')
8      if marker and platform.system() != 'Linux':
9          pytest.skip('N/A on {}'.format(platform.system()))
```

Параметризация фикстур

Фикстуры можно параметризовать. Параметризованные фикстуры можно комбинировать.

```
1  @pytest.fixture(params=['ora', 'pg', 'sqlite'])
2  def dburi(request):
3      return create_db_uri(request.param)
4
5  @pytest.fixture(params=['ipv4', 'ipv6'])
6  def addr_family(request):
7      return socket.AF_INET if request.param == 'ipv4' else
   ↪   socket.AF_INET6
8
9  def test_txn(dburi):
10     inst = MyObj(dburi)
11     assert inst.transaction_works()
12
13 def test_conn(dburi, addr_family):
14     inst = MyObj(dburi, addr_family)
15     assert inst.it_works()
```

Применять skip можно и на этапе параметризации:

```
try:
    import cx_Oracle as ora
except ImportError:
    ora = None

needs_ora = pytest.mark.skipif(ora is None, reason='No Oracle
 ↪  installed')

@pytest.fixture(params=['pg', needs_ora('ora')])
def dburi(request):
    return create_db_uri(request.param)

@needs_ora
def test_one():
    pass

def test_two(dburi):
    pass
```

Всё вместе

- Фикстуры в pytest — функции.

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.
- Teardown: `request.addfinalizer` или используя декоратор `yield_fixture`.

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.
- Teardown: `request.addfinalizer` или используя декоратор `yield_fixture`.
- Кэширование. Используя параметр `scope` (function, class, module, session).

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.
- Teardown: `request.addfinalizer` или используя декоратор `yield_fixture`.
- Кэширование. Используя параметр `scope` (function, class, module, session).
- Модульность. Фикстуры используют фикстуры (внимательно с кешированием!).

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.
- Teardown: `request.addfinalizer` или используя декоратор `yield_fixture`.
- Кэширование. Используя параметр `scope` (function, class, module, session).
- Модульность. Фикстуры используют фикстуры (внимательно с кешированием!).
- Могут вызвать skip или fail всех зависимых тестов.

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.
- Teardown: `request.addfinalizer` или используя декоратор `yield_fixture`.
- Кэширование. Используя параметр `scope` (function, class, module, session).
- Модульность. Фикстуры используют фикстуры (внимательно с кешированием!).
- Могут вызвать skip или fail всех зависимых тестов.
- Полная интроспекция контекста в котором вызывается фикстура (функция, класс, модуль) через `request`. К примеру, возможно получить список маркеров.

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.
- Teardown: `request.addfinalizer` или используя декоратор `yield_fixture`.
- Кэширование. Используя параметр `scope` (function, class, module, session).
- Модульность. Фикстуры используют фикстуры (внимательно с кешированием!).
- Могут вызвать skip или fail всех зависимых тестов.
- Полная интроспекция контекста в котором вызывается фикстура (функция, класс, модуль) через `request`. К примеру, возможно получить список маркеров.
- Глобальные фикстуры (autouse).

Всё вместе

- Фикстуры в pytest — функции.
- Могут быть определены: на уровне класса, на уровне модуля и в conftest.
- Teardown: `request.addfinalizer` или используя декоратор `yield_fixture`.
- Кэширование. Используя параметр `scope` (function, class, module, session).
- Модульность. Фикстуры используют фикстуры (внимательно с кешированием!).
- Могут вызвать skip или fail всех зависимых тестов.
- Полная интроспекция контекста в котором вызывается фикстура (функция, класс, модуль) через `request`. К примеру, возможно получить список маркеров.
- Глобальные фикстуры (autouse).
- Параметризация.

Демонстрация

- Рассмотрим применение фикстур на примере несложных UI тестов.

Демонстрация

- Рассмотрим применение фикстур на примере несложных UI тестов.
- Сценарий для автоматизации:

Демонстрация

- Рассмотрим применение фикстур на примере несложных UI тестов.
- Сценарий для автоматизации:
- – Открыть браузер на странице `http://sputnik.ru/search`.

Демонстрация

- Рассмотрим применение фикстур на примере несложных UI тестов.
- Сценарий для автоматизации:
- – Открыть браузер на странице `http://sputnik.ru/search`.
- – Ввести запрос в поле и нажать ENTER.

Демонстрация

- Рассмотрим применение фикстур на примере несложных UI тестов.
- Сценарий для автоматизации:
- – Открыть браузер на странице `http://sputnik.ru/search`.
- – Ввести запрос в поле и нажать ENTER.
- – Сделать необходимую проверку.

Демонстрация

- Рассмотрим применение фикстур на примере несложных UI тестов.
- Сценарий для автоматизации:
- – Открыть браузер на странице `http://sputnik.ru/search`.
- – Ввести запрос в поле и нажать ENTER.
- – Сделать необходимую проверку.
- Использовать будем:

Демонстрация

- Рассмотрим применение фикстур на примере несложных UI тестов.
- Сценарий для автоматизации:
- – Открыть браузер на странице `http://sputnik.ru/search`.
- – Ввести запрос в поле и нажать ENTER.
- – Сделать необходимую проверку.
- Использовать будем:
- Pytest (что же еще?), Selenium Webdriver (автоматизация браузера), Virtual Display (виртуальный дисплей, без иксов), Allure Framework (для отчетов).

Полезные ресурсы

- Docs: http://docs.pytest.org/en/latest/fixture.html
- “Advanced py.test fixtures”: https://youtu.be/IBC_dxr-4ps and slides
- “Improve your testing with Pytest and Mock”: https://youtu.be/RcN26hznmk4