Нескучное тестирование с pytest

Роман Иманкулов / @rdotpy / 14 июня 2014

http://www.devconf.ru

Кратко о докладчике

- Жизненный путь – с 1983
- Python – с 2005
- Server-Side для веб – с 2006
- Разработка с pytest – с 2012

TDD в Python — это религия

- Самоуничижение
- Очищение через страдание
- Мистический опыт

Самоуничижение. Первородный грех

- Врожденные пороки — нестрогая типизация и duck typing
- Как следствие — природная склонность программиста на Python к совершению маленьких и глупых ошибок

Очищение через страдание

Boilerplate Code

```
class TestSequenceFunctions(unittest.TestCase):
    def setUp(self):
        ...
    def tearDown(self):
        ...
    def testFoo(self):
        ...
```

Очищение через страдание

Многословные ассерты

```
self.assertEqual(foo, 1,
                 'foo is not equal to one')
```

Мистический опыт

Django testing setups & teardowns

Есть ли альтернатива?

pytest

pytest — это не еще один xUnit фреймворк!

То, что отличает pytest от других фреймворков

pytest fixtures

pytest fixtures

Наивный подход. Как это бы сделал я сам

file: fixtures.py

```
def get_user():
    return User(name='Roman', age=30, ...)
```

file: test_user.py

```
def test_user():
    user = get_user()
    assert user.name == 'Roman'
```

pytest fixtures

Подход pytest

file: conftest.py

```
@pytest.fixture
def user():
    return User(name='Roman', age=30, ...)
```

file: test_user.py

```
def test_user(user):
    assert user.name == 'Roman'
```

Зависимости между fixtures

```
@pytest.fixture
def user():
    return User(name='Roman', age=30, ...)

@pytest.fixture
def task(user):
    return Task(user=user, name='...')

def test_task(task):
    assert task.user.name == 'Roman'
```

Fixture dependencies. Patching object

```
@pytest.fixture
def premium(user)
    user.set_premium()


def test_premium(user, premium):
    assert user.is_premum()
```

yield_fixture

setup и teardown в одном флаконе

```
@pytest.yield_fixture
def user():
    obj = User(name='Roman', age=30, ...)
    yield obj
    obj.delete()
```

Fixture scopes

- function scope
- module scope
- session scope

Session fixture. Локальный кеш

```
@pytest.yield_fixture(scope='session', autouse=True)
def local_cache():
    old_settings = settings.CACHES
    settings.CACHES = {'default': {…}}
    yield
    settings.CACHES = old_settings
```

Function fixture. Database transaction rollback

```
@pytest.yield_fixture
def tx():
    db().start_transaction()
    yield
    db().rollback()


def test_user(user, tx, project, task):
    # project & task will be removed automatically
```

Session fixture. Чистый redis

```
@pytest.yield_fixture(scope='session')
def redis_server():
    proc = subp.Popen(['redis-server', '--port',
7777], ... )
    yield proc
    proc.terminate()

@pytest.fixture
def rc(redis_server):
    client =
redis.StrictRedis('redis://127.0.0.1:7777')
    client.flushall()
    return client
```

fixtures parametrization

Функция возвращает функцию

```
@pytest.fixture
def set_lang(user):
    def func(lang_code):
        user.set_lang(lang_code)
    return func

def test_languages(user, set_lang):
    set_lang('ru')
    ...
```

Странные вещи

Fixtures в отдельном потоке. http://bit.ly/test_pool

```
@pytest.fixture(scope='session')
def item_gen():
    gen = Generator(lambda: .)
    gen.start()
    return gen
```

```
@pytest.fixture(scope='session')
def item_rel():
    rel = Releaser(lambda o: ...)
    rel.start()
    return rel
```

```
@pytest.yield_fixture
def item(item_gen, item_rel):
    item = item_gen.get()
    yield item
    item_rel.put(item)
```

Как ещё использовать fixtures

- **warnings**: turn MySQL warnings to errors
- **mock:** подготовка mockup объектов
- **freezegun:** управление временем
- **selenium:** запуск веб-драйвера

Как ещё использовать fixtures

- Проверка корректности settings перед тестом
- Залогиненный тестовый http client для Flask или Django

О чём я ещё не рассказал

- **def pytest_addoption():** параметры командной строки
- **@pytest.mark.parametrize:** параметризация тестов

- **pytest-django:** интеграция с Django
- **pytest-xdist:** параллельные и распределенные тесты

- **tox:** выполнение тестов для разных python
- **detox:** то же самое, только параллельно

Спасибо! Вопросы?

Роман Иманкулов / @rdotpy / http://imankulov.name