mockery

George Y. Kussumoto

contato@georgeyk.com.br
georgeyk @ github
georgeyk0 @ twitter

jan/2018

TOC

olist

- Revisão de conceitos: testes
- unittest.mock
  - parte 1 - básico
  - parte 2 - “avançado”
- slides & exemplos:
  - https://github.com/georgeyk/umbrella/tree/master/mockery

Revisão: tests

Revisão: testes

olist

A unit test exercises the smallest piece of testable software in the application to determine whether it behaves as expected.

M. Fowler

Revisão: testes

olist

regressão

refactoring

documentation

design / quality

risco

velocidade

custos

Revisão: testes

olist

unit test

integration test

system test

collaboration test

acceptance test

contractual test

functional test

exploratory test

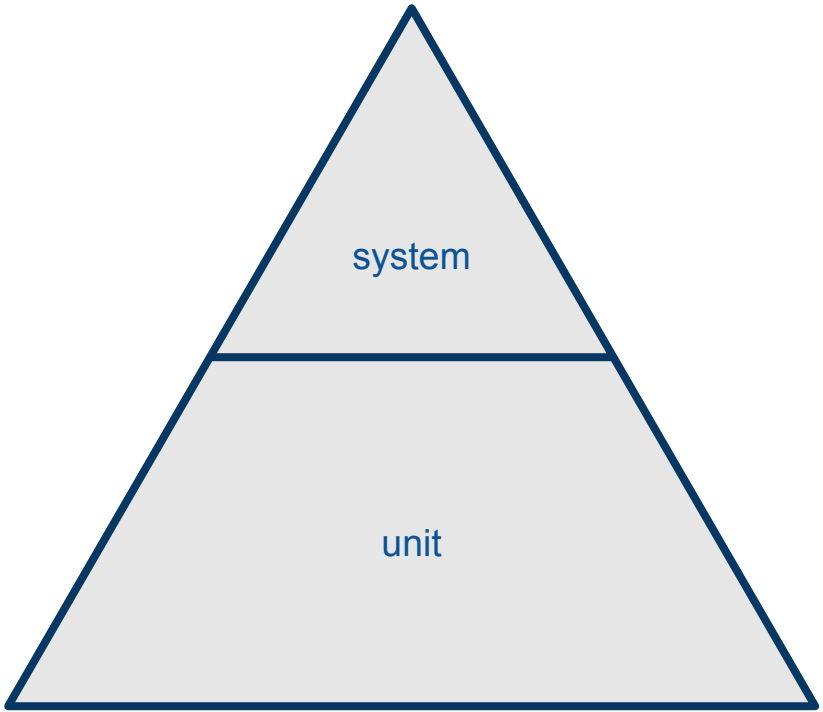
end-to-end test

...

Revisão: testes

olist

abstração



deps ++
complexos
lentos
cobertura --
20% ~

deps --
rápidos
granulares
isolados
cobertura ++
80% ~

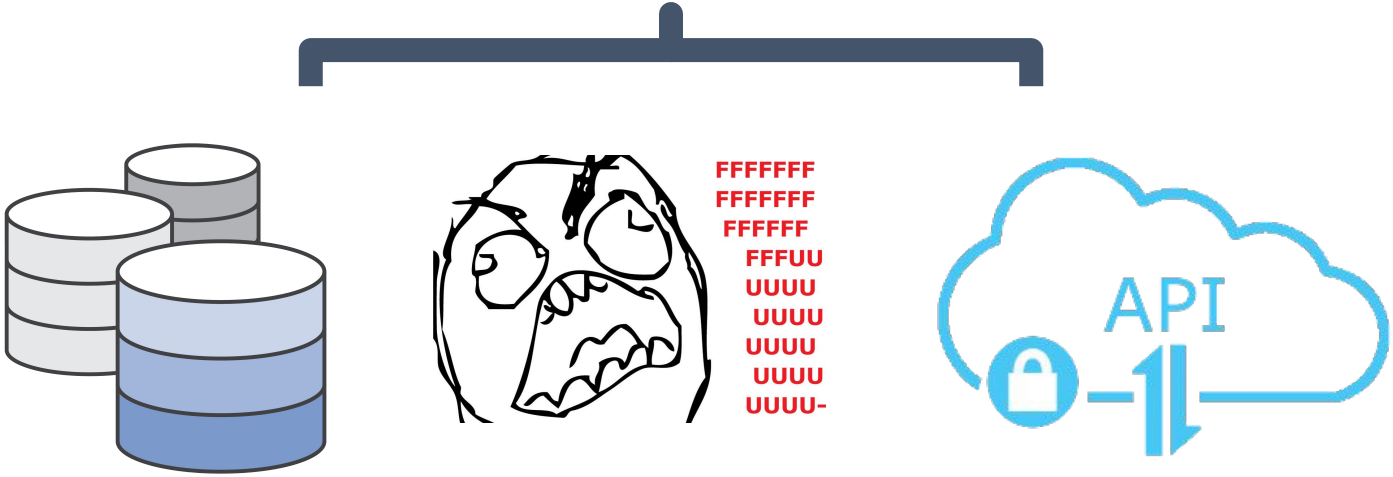
Revisão: testes

olist

```
def test_person_name_capitalized(address_data):
    address = Address(address_data)
    person = Person(name='foobar', address=address)
    assert person.name == 'Foobar'
```

Revisão: testes

```
person = Person(name='foobar', address=address)
```



parte 1: mocks

parte 1: mocks

olist

Test doubles

- Dummy objects
- Fake
- Stubs (fixtures)
- Spies
- Mocks
  - “VCRpy” (http)

parte 1: mocks

olist

umbrella/mockery/mock_101.py

parte 1: mocks

olist

```
# umbrella/mockery/where_to_patch/ex0.py
>>> with patch('sys.stdout') as mock_stdout:
...     print('hello')
...     assert mock_stdout.write.called
...

>>> print('world')
world
```

parte 1: mocks

olist

Aplicando “patch” (umbrella/mockery/where_to_patch/*)

```
def get_number():
    return ...
```

a.py

```
import a

class UserInput:
    def read_even_number(self):
        number = a.get_number()
        ...
```

b.py

```
from b import UserInput


@patch('?.get_number')
def test_user_input_even(mock_get_number):
    mock_get_number.return_value = 2
    b = UserInput()

    assert b.read_even_number() == 2
    mock_get_number.assert_called_once_with()
```

test_b.py

parte 1: mocks

olist

Aplicando “patch” (umbrella/mockery/where_to_patch/*)

```
def get_number():
    return ...
```

a.py

```
from a import get_number

class UserInput:
    def read_even_number(self):
        number = get_number()
        ...
```

b.py

@patch(‘**a**.get_number’) -> mock_get_number

- altera a referência da **definição** da função

@patch(‘**b**.get_number’) -> mock_get_number

- altera a referência na **“busca”** da função

parte 1: mocks

olist

Aplicando “patch” (umbrella/mockery/where_to_patch/*)

- namespaces
- ex1_definition.py
- ex1_lookup.py

- c.py
- test_c.py

```
from b import UserInput


@patch('b.get_number')
def test_user_input_even(mock_get_number):
    mock_get_number.return_value = 2
    b = UserInput()

    assert b.read_even_number() == 2
    mock_get_number.assert_called_once_with()
```

test_b.py

parte 2: mocks

parte 2: mocks

olist

umbrella/mockery/mock_bad.py

parte 2: mocks

olist

umbrella/mockery/mock_good.py

parte 2: mocks

olist

“mockou”

parte 2: mocks

olist

Arquitetura / Design feedback

- teste difícil de ser escrito
- muitas dependências ou asserts
- efeito cascata / suite “binária” de testes
- estratégia de refatoração
- umbrella/mockery/mock_feedback.py

bônus: vcr

olist

Bônus: vcr (https://github.com/kevin1024/vcrpy)

- prototipação rápida e fiel
- rápido e fácil de gerar* (cassettes)
- pode expor informações indesejadas
- dependência de estado
- difícil de manter
- testes “ruins”

bônus: coverage

olist

Bônus: coverage (https://bitbucket.org/ned/coveragepy)

- 100 % de cobertura != 100% testado
- baixa cobertura % -> deficiente
- qualidade >>> quantidade
- casos “ocultos”
- umbrella/mockery/coverage/*

recap

olist

Resumo:

- testing review
- mocks: why / when
- mocks: what / how
- mocks: where
- mocks: “fixing”

Dúvidas ?

olist | venda nos MAIORES

References

olist

- **Fast Test, Slow Test - Gary Bernhardt -** https://www.youtube.com/watch?v=RAxiiRPHS9k
- **Boundaries - Gary Bernhardt -** https://www.youtube.com/watch?v=yTkzNHF6rMs
- **Microservice Testing - Martin Fowler -** https://martinfowler.com/articles/microservice-testing/
- **Integrated Tests Are A Scam - Rainsberger -** https://www.youtube.com/watch?v=VDfX44fZoMc
- **Why our code smells - Brandon Keepers -** https://speakerdeck.com/bkeepers/why-our-code-smells

References

olist

- **unittest.mock -** https://docs.python.org/3/library/unittest.mock.html
- **Mocks Aren't Stubs - Martin Fowler -**

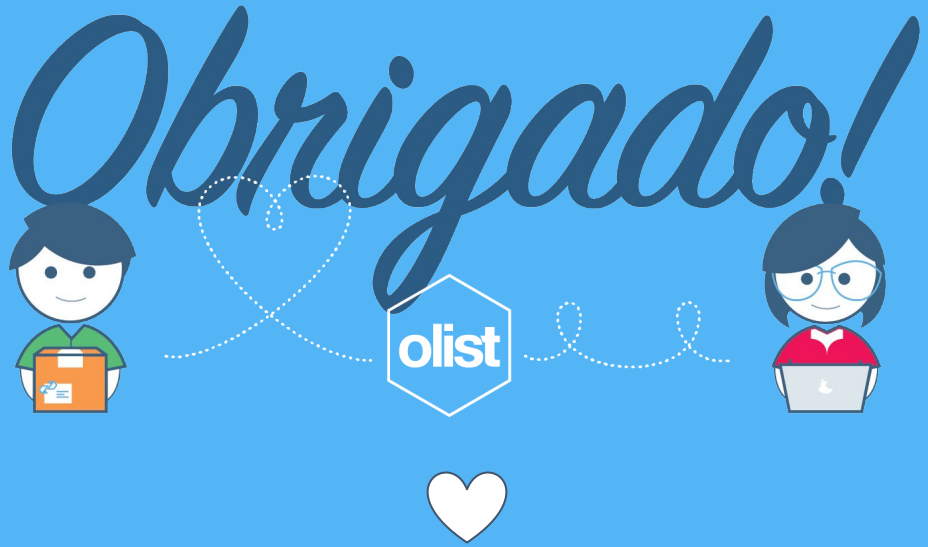
  https://martinfowler.com/articles/mocksArentStubs.html
- **Test Isolation Is About Avoiding Mocks - Gary Bernhardt -**

  https://www.destroyallsoftware.com/blog/2014/test-isolation-is-about-avoiding-mocks
- **Test-induced design damage - DHH -**

  http://david.heinemeierhansson.com/2014/test-induced-design-damage.html

Feedback

olist



contato@georgeyk.com.br
georgeyk @ github
georgeyk0 @ twitter

olist.com