ТЕХНОТРЕК

Лекция 2

Тестирование Web-приложений

Елизарова Алена

Виды тестирования

- Unit-тестирование
- Функциональное тестирование
- Интеграционное тестирование
- Нагрузочное тестирование

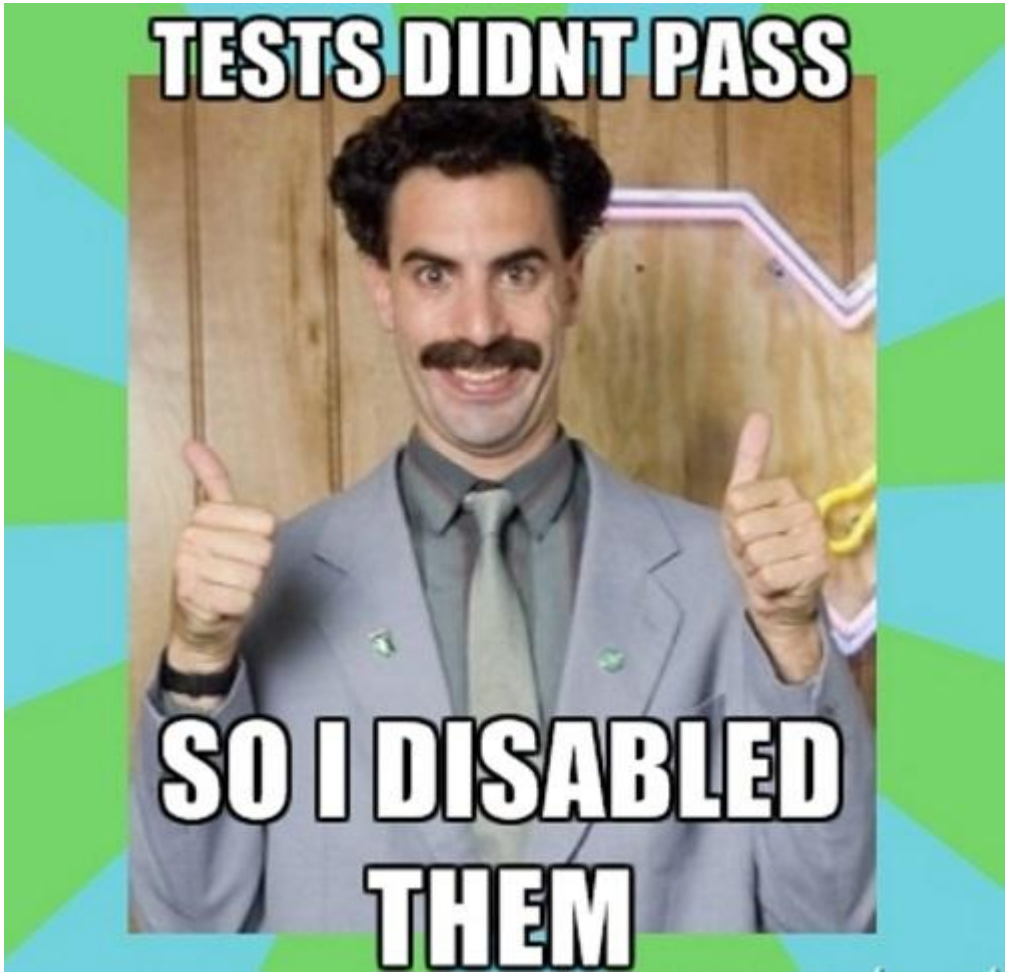
- Client side (selenium)
- Server side

TDD - test driven development - техника разработки ПО, основывается на повторении коротких циклов разработки: пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода

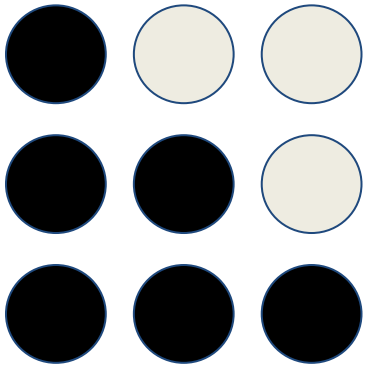
Цели тестирования

- Проверка правильности реализации
- Проверка обработки внештатных ситуаций и граничных условий
- Минимизация последствий

Цели тестирования



Степень покрытия тестами (test coverage)



Лендинговая страница

Web-приложение с бизнес-логикой

Фреймворки и библиотеки

coverage - библиотека для проверки покрытия тестами

```
pip install coverage
coverage run ./manage.py test blog
```

Инструменты для тестирования в Python

- doctest
- pytest
- hypothesis
- unittest

doctest

```
'''
Module showing how doctests can be included with source code
Each '>>>' line is run as if in a python shell, and counts as a
test.
The next line, if not '>>>' is the expected output of the previous
line.
If anything doesn't match exactly (including trailing spaces), the
test fails.
'''

def multiply(a, b):
    """
    >>> multiply(4, 3)
    12
    >>> multiply('a', 3)
    'aaa'
    """
    return a * b
```

Запуск

```
python -m doctest <file>
```

pytest

```
class TestClass(object):
    def test_one(self):
        x = "this"
        assert 'h' in x

    def test_two(self):
        x = "hello"
        assert hasattr(x, 'check')
```

Запуск

```
pytest <file>
```

hypothesis

```
from hypothesis import given
from hypothesis.strategies import text

@given(text())
def test_decode_inverts_encode(s):
    assert decode(encode(s)) == s
```

Тестирование в Django. unittest

```
from django.test.client import Client
```

Тестовый http-клиент. Имитирует работу браузера, может отправлять http-запросы, сохраняет cookies между вызовами

Запрашивает относительный путь, например /login/

Тестирование в Django. unittest

```
import json
from django.test import TestCase
from django.test.client import Client

from video.models import Category

class VideoAPITest(TestCase):

    def setUp(self):
        self.client = Client()
        self.category = Category.objects.create(name='test category')
        self.response_map = {
            'name': 'test category',
            'video_count': 0
        }

    def test_category_values(self):
        response = self.client.get('/api/v1/video_category/')
        content = json.loads(response.content)

        for key, value in self.response_map.iteritems():
            self.assertEqual(
                content['objects'][0].get(key),
                value,
                msg='{} not equal'.format(key)
            )

    def tearDown(self):
        print ('I'm done')
```

unittest. Расширенный набор проверок assert

```
assertEqual(a, b)
assertNotEqual(a, b)
assertTrue(x)
assertFalse(x)
assertIs(a, b)
assertIsNot(a, b)
assertIsNone(x)
assertIn(a, b)
assertIsInstance(a, b)
assertLessEqual(a, b)
assertListEqual(a, b)
assertDictEqual(a, b)
assertRaises(exc, fun, *args, **kwds)
```

mock

Mock - подменяет объекты (функции, классы) на так называемые mock-объекты, заглушки.

```
class TestUserSubscription(TestCase):

    @patch('users.views.get_subscription_status')
        def test_subscription(self, get_subscription_status_mock):
            get_subscription_status_mock.return_value = True

            ...
```

mock

Атрибуты объекта Mock с информацией о вызовах

```
called – вызывался ли объект вообще
call_count – количество вызовов
call_args – аргументы последнего вызова
call_args_list – список всех аргументов
method_calls – аргументы обращений к вложенным методам и атрибутам
mock_calls – то же самое, но в целом и для самого объекта, и для
вложенных
```

```
# пример

self.assertEqual(get_subscription_status_mock.call_count, 1)
```

factory_boy

Библиотека factory_boy служит для генерации тестовых объектов (в т.ч. связанных) по заданным параметрам.

```
# объявляем фабрику
class RandomUserFactory(factory.Factory):
    class Meta:
        model = models.User

    first_name = factory.Faker('first_name')
    last_name = factory.Faker('last_name')
    email = factory.Sequence(lambda n: 'person{0}@example.com'.format(n))
```

```
# Returns a User instance that's not saved
user = RandomUserFactory.build()

# Returns a saved User instance
user = RandomUserFactory.create()

# Returns a stub object (just a bunch of attributes)
obj = RandomUserFactory.stub()
```

Запуск тестов

```
# тестирование всего проекта
./manage.py test

# тестирование указанных через пробел приложений
./manage.py test app_name1 app_name2

# тестирование одного кейса - набора тестов
./manage.py test app_name.tests.SomeTestCase

# тестирование одно метода
./manage.py test app_name.tests.SomeTestCase.test_some_method
```

Оптимизация. Тестовое окружение

```
# настроить verbose
./manage.py test -v 2

# не запускать миграции при каждом запуске тестов
./manage.py test --keepdb

# распараллелить запуск на несколько БД
./manage.py test --parallel=2

# запустить тесты для быстрого обнаружения ошибок
./manage.py test --failfast
```

Использовать легковесную БД при тестировании (если возможно), переопределив настройки settings.py

```
TESTING = 'test' in sys.argv

if TESTING:
    DATABASES['default'] = {'ENGINE': 'django.db.backends.sqlite3'}
```

Фикстуры

```
[
  {
    "model": "myapp.person",
    "pk": 1,
    "fields": {
      "first_name": "John",
      "last_name": "Lennon"
    }
  },
  {
    "model": "myapp.person",
    "pk": 2,
    "fields": {
      "first_name": "Paul",
      "last_name": "McCartney"
    }
  }
]
```

```
django-admin dumpdata app_name mydata.json # создать фикстуру
django-admin loaddata mydata.json # загрузить фикстуру
```

Домашнее задание №2

- Написать тесты для проверки правильности ответа сервера (status code) - 1
- Написать тесты на формат возвращаемых значений API с заглушками (mock, factory_boy) - 4
- Написать тесты на взаимодействие с базой данных, создав некоторые объекты в setUp - 4
- Подготовить фикстуры данных для тестирования - 2
- Узнать степень покрытия тестами - 1

Срок сдачи

6 октября 2018

ТЕХНОТРЕК

Спасибо за внимание!