

TDD em Django

Sem Desculpas

adriano petrich
petrich@gmail.com
#fis11

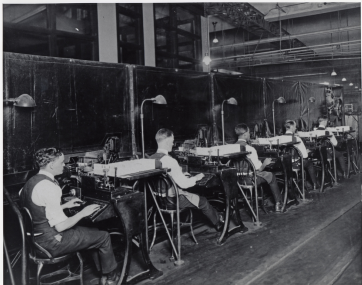
Código sem testes é código já quebrado quando foi planejado

-- Jacob Kaplan-Moss

um dos criadores do django

Estamos em 2010

2010 > 1960



2010 > 1999



**Alguma buzzword "ágil"
você tem que estar
usando**

Buzzwords são quintessenciais

Buzzwords trazem sinergia viral e o empowerment das melhores práticas para a cauda longa

3 anos atrás

equipe: >40 pessoas numa mesma sala

escopo: Webapp em Tomcat

buzzwords: Bodyshop típico: PMI, cmms.. (< 1999)

2.5 anos atrás

equipe: 5 pessoas espalhadas pelo mundo

escopo: Modificações no nível de uma distro

buzzwords: Scrum, cultura de testes, sprints, entregas semanais

5 > 40

Metodologias ágeis:

Extreme Programming(XP)

Scrum

Kanban

Feature Driven Development (FDD)

Práticas ágeis:

Test Driven Development (TDD)

Behavior Driven Development (BDD)

Code refactoring

Continuous Integration

Pair Programming

Planning poker

TDD

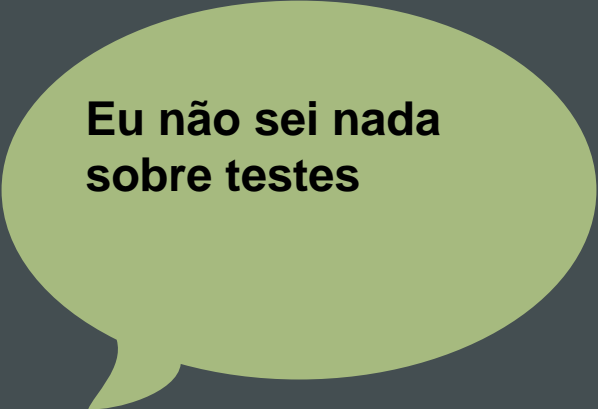
Sustentável

Fácil

Não depende da gerência

**TDD não é difícil.
Difícil é não fazer
quando voce
acostuma**

Então, chega de desculpas:



**Eu não sei nada
sobre testes**

O ecossistema de testes no python

- Tipos
- Sabores
- TestRunners

Tipos de testes

Doctest

```
def add(a,b):  
    """  
    testa a soma  
    >>> add(1,2)  
    3  
    """  
    return a + b
```

Unittest

`unittest.TestCase`

`django.test.TestCase`

django.test.TestCase

```
from django.test import TestCase

class SimpleTest(TestCase):
    def test_adicao(self):
        """
        Testa que a adicao de 1 + 1 da 2.
        """
        self.assertEqual(1 + 1, 2)
```

Sabores de testes

Unitários

Nível de função

```
self.assertTrue(add(1,2),3)
```

Integração

Entre Módulos

```
r = self.client.get('/foo')
self.assertRedirects(r, '/login/')
self.client.login(user_name='foo'
                  ,password='bar')
r = self.client.get('/foo')
self.assertEqual(r.status_code, 200)
```


De Regressão

Correção de erros

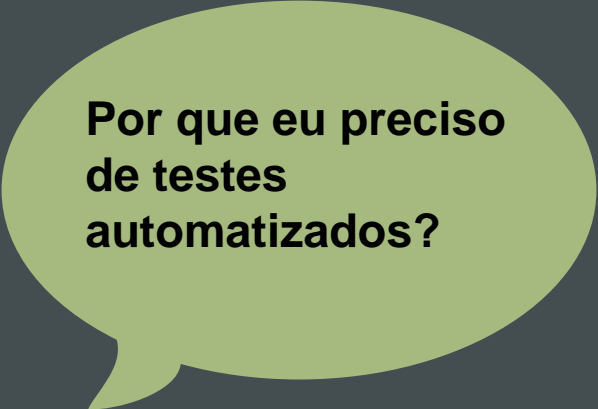
TestRunners

Acha e Roda os testes

- Padrão
- py.test
- nose
- outros

Meu estilo

- Django.test.TestCase
- **Unitário**
Um TestCase por modelo
Um ou mais testes por função
- **Integração**
Um por TestCase por conjunto de apps
- **Regressão**
Um teste por erro
- **nose / django-nose**
Acha testes



**Por que eu preciso
de testes
automatizados?**

Código evolve

**Se o seu código não
tem testes refatorar ele
é um pesadelo**

Imagina isso

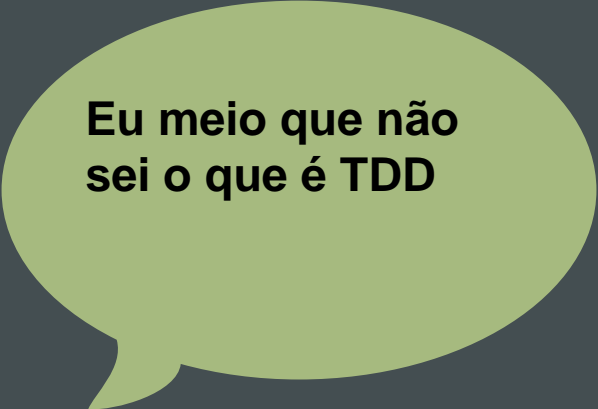
```
$ cat `find . | grep "py$" \
    | grep -v migration` | wc -l
47260
```


Agora isso:

```
$ cat `find . | grep "py$" \  
      | grep test` | wc -l  
34108
```

Tranquilidade de refatorar

Felicidade é um código com boa cobertura



**Eu meio que não
sei o que é TDD**

**Ciência da computação
é tanto sobre
computadores quanto
como a astronomia é
sobre telescópios**

-- E W Dijkstra

**Test Driven
Development é tanto
sobre testes assim
quanto a ciência da
computação é sobre
computadores**

**TDD é sobre
desenvolvimento e
qualidade**

**Testes são um
subproduto**

TDD

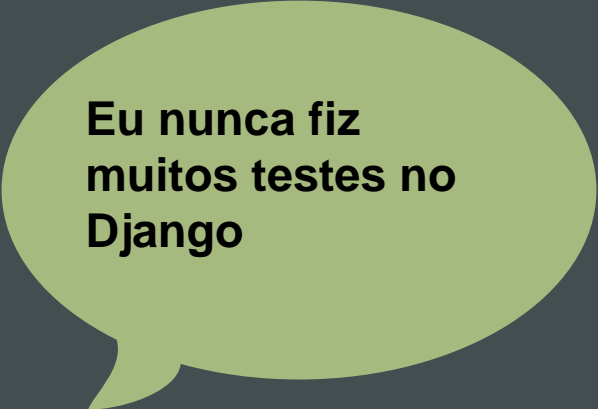
TDD

Só escreve **código** quando testes falham

TDD

Só escreve **código** quando testes falham

Só escreve **testes** quando testes passam



**Eu nunca fiz
muitos testes no
Django**

Como fazer

```
$ django-admin.py startproject foobar  
$ cd foobar/  
$ chmod +x manage.py  
$ vi settings.py
```

settings.py

```
import os
PROJECT_PATH = os.path.abspath(
    os.path.split(__file__)[0])
...
configura o banco
...
TEMPLATE_DIRS = (
    os.path.join(PROJECT_PATH, 'templates'),
)
```

Hora de testar

```
./manage.py test
```

```
-----
```

```
Ran 0 tests in 0.000s
```

```
OK
```

```
Destroying test database 'default'...
```

TDD

Só escreve **código** quando testes falham

Só escreve **testes** quando testes passam

Passou

Escreve testes

Mais Testes, então

```
./manage.py startapp forum  
cd forum/
```

Meu estilo (v.2)

```
rm tests.py  
mkdir tests  
touch tests/__init__.py  
touch tests/test_models.py
```

vi tests/test_models.py

```
#coding:utf8  
from django.test import TestCase  
  
class TopicoTest(TestCase):
```

Teste de importação

```
def test_existe(self):  
    """ O topico esta la? """  
    try:  
        from foobar.forum.models import Topico  
    except ImportError:  
        self.fail('Nao existe topico')
```

Inclui a app no projeto

```
INSTALLED_APPS = (  
    ...  
    'foobar.forum',  
)
```

Testa

```
./manage.py test
```

```
-----
```

```
Ran 0 tests in 0.000s
```

```
OK
```

```
Destroying test database 'default'...
```

TDD sem desculpas

0 testes!

nose

Acha testes para você sem que você tenha que por eles no `__init__.py`

Dá pra chamar o pdb no ponto em que falha (`--pdb-failures`) (ou `ipdb`)

django-nose

```
$ pip install nose  
$ pip install django-nose
```

settings.py

```
TEST_RUNNER = 'django_nose.NoseTestSuiteRunner'

INSTALLED_APPS = (
    ...
    'south', # migracoes
    'django_nose', # depois do south
)
#opcional
#NOSE_ARGS = ['--pdb-failures', ]
```

Testa de novo

```
F
=====
FAIL: O topico esta la?
-----
Traceback (most recent call last):
  File "test_models", line 18, in test_existe
    self.fail('Nao existe topico')
AssertionError: Nao existe topico
-----
Ran 1 test in 0.003s
```

TDD

Só escreve **código** quando testes falham

Só escreve **testes** quando testes passam

Falhou

Escreve código

vi forum/models.py

```
class Topico(models.Model):  
    """representa um topico"""  
    pass
```

testa

.

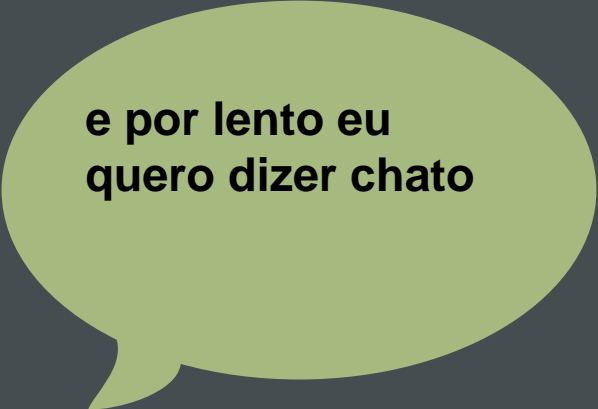
Ran 1 test in 0.014s

Pera!

Voce gastou 8 slides para escrever um pass?



**Mas TDD é muito
lento**



**e por lento eu
quero dizer chato**

**A primeira vez é
lento**

Entenda o que você está testando

```
try:
    from foobar.forum.models import Topico
except ImportError:
    self.fail('Nao existe topico')
```

Não teste a framework

Testa a **lógica da sua** aplicação

Facilitadores

Continuous testing

Toda vez que você salva um arquivo ele rerola os testes

django test extensions

Faz isso para você

Ainda é um pouco tosco

django-test-extensions

```
$ pip install django-test-extensions
```

settings.py

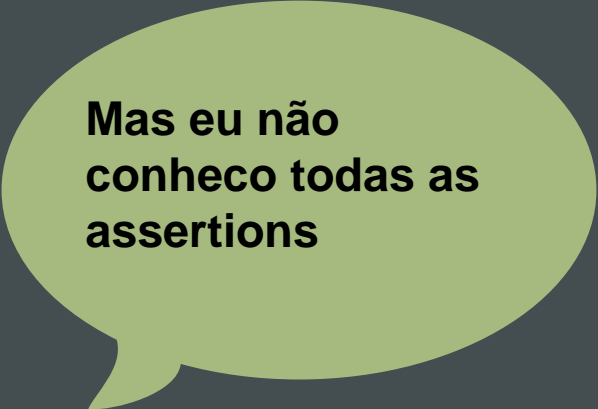
```
INSTALLED_APPS = (  
    ...  
    'south', # migracoes  
    'django_nose', # depois do south  
    'test_extensions', # depois do south  
)
```

Rodando o servidor

```
$ ./manage.py runtester
```

ou ainda

```
$ ./manage.py runtester forum
```



**Mas eu não
conheço todas as
assertions**

Bico

Modo mais fácil:

no ./manage shell (com ipython instalado)

```
>>> from django.test import TestCase  
>>> In [2]: TestCase.assert<tab><tab>
```

asserts

<code>TestCase.assert_</code>	<code>TestCase.assertAlmostEqual</code>
<code>TestCase.assertAlmostEquals</code>	<code>TestCase.assertContains</code>
<code>TestCase.assertEqual</code>	<code>TestCase.assertEquals</code>
<code>TestCase.assertFalse</code>	<code>TestCase.assertFormError</code>
<code>TestCase.assertNotAlmostEquals</code>	<code>TestCase.assertNotContains</code>
<code>TestCase.assertNotEqual</code>	<code>TestCase.assertNotEquals</code>
<code>TestCase.assertRaises</code>	<code>TestCase.assertRedirects</code>
<code>TestCase.assertTemplateNotUsed</code>	<code>TestCase.assertTemplateUsed</code>
<code>TestCase.assertTrue</code>	<code>TestCase.assertNotAlmostEqual</code>

Asserts básicas

Essas você deve usar bastante

```
assertTrue(True)
```

```
assertFalse(False)
```

```
assertEqual(1,1)
```

```
assertNotEqual(1,2)
```


Asserts amigáveis

Essas facilitam a vida para testes funcionais

```
assertContains(response, texto, status)  
assertNotContains(response, texto, status)
```

exemplo

```
def test_welcome(self):  
    resp = self.client.get('/', {})  
    self.assertContains(resp, '<h1>Oi</h1>'  
                        , 200)
```

Asserts amigáveis (cont)

```
assertRedirects(response,nova_url)
assertTemplateUsed(response,template)
assertTemplateNotUsed(response,template)
assertFormError(response,form,fields,errors)
```

WTF?

```
assertAlmostEqual
```

```
assertNotAlmostEqual
```

Não quase iguais?

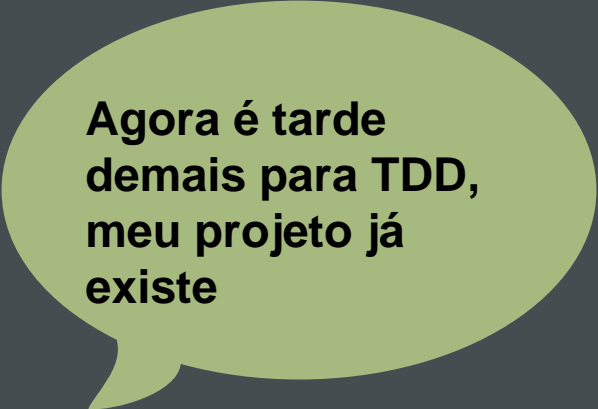
```
a = 1.21
b = 1.22
#sao iguais ate a primeira casa
self.assertEqual(a,b,1)
#diferentes depois da segunda casa
self.assertNotAlmostEqual(a,b,2)
```

Asserts que eu não uso

```
assertRaises
```

Testo assim:

```
try:
    foobar.bang():
    self.fail('Bang tem que explodir')
except ExplodingException:
    pass
```



**Agora é tarde
demais para TDD,
meu projeto já
existe**

Pera! Olha só

- Testes de Regressão
- `django_test_utils`

Seu melhor amigo

Garante que um erro que aconteceu nunca mais volte a acontecer

Usado por todos os grandes projetos de software livre

Mesmo você não vai fazer mais nenhuma forma de teste você tem que fazer esta

Testes de Regressão

Encontrou um erro

```
[24/Jul/2010 11:14:51] "GET / HTTP/1.1" 404 1946
```

Escreve um teste que falha por causa do erro

```
$ vi forum/test_regression.py
```

cont

```
#coding:utf8
from django.test import TestCase

class TestRegression(TestCase):
    """testes de regressao"""
```

cont+=1

```
def test_regress_home(self):  
    """Home precisa existir"""  
    r = self.client.get('/', {})  
    self.assertEqual(r.status_code, 200)
```

Testa e falha

```
..E
=====
ERROR: Home precisa existir
-----
Traceback (most recent call last):
  File "foobar/forum/tests/test_regresssion.py",
        line 10, in test_regress_home
    r = self.client.get('/', {})
    ...
    raise TemplateDoesNotExist(name)
TemplateDoesNotExist: 404.html
```


Corrige o erro

```
from django.views.generic.simple import direct_to_template
urlpatterns = patterns('',
    ...
    (r'^$', direct_to_template, {'template': 'index.html'}),
    ...
)
```

```
$ vi templates/index.html
```

Roda os testes e passa

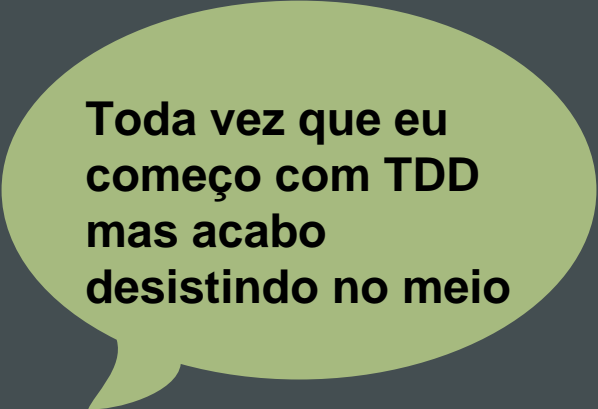
```
nosetests --verbosity 1
```

```
....
```

```
-----  
Ran 4 tests in 0.025s
```

```
OK
```

**Garantia que erros
antigos não vão
retornar para te
assombrar**



**Toda vez que eu
começo com TDD
mas acabo
desistindo no meio**

**2 formas
sustentáveis para
começar e
continuar com
TDD**

Primeiro:

TDD:Eu queria ter isso

Você escreve nos testes a API que você queria ter

Eu queria que fosse assim:

```
def test_metodos(self):  
    topico = Topico()  
    self.assertTrue(hasattr(topico, 'titulo'))  
    self.assertTrue(hasattr(topico, 'replies'))
```


Testa

```
F.  
=====  
FAIL: test_metodos (test_forum.TestForum)  
-----  
Traceback (most recent call last):  
    self.assertTrue(hasattr(topico, 'titulo'))  
AssertionError  
  
-----  
Ran 2 tests in 0.002s  
FAILED (failures=1)
```

Implementa

```
class Topico(models.Model):  
    """representa um topico"""  
    titulo = models.CharField(max_length=64)  
class Resposta(models.Model):  
    '''Uma resposta no topico'''  
    topico = models.ForeignKey(Topico,  
                               related_name='replies')
```

Testa

```
..
```

```
-----  
Ran 2 tests in 0.002s
```

```
OK
```

Prós e Cons

- Não é exatamente TDD
- Funciona
- Mais rápido
- Você está perdendo cobertura

Segundo: SDT

SDT

Eu não faço TDD eu faco Stupidity-driven testing. Quando eu faco algo estúpido, eu escrevo um teste para garantir que eu não vou repetir isso de novo

--Titus Brown pycon '07

Em suma

Escreve código para solucionar um problema

Se o código quebrar de alguma forma besta

Escreve um teste para isso nunca vai acontecer de novo

goto 10

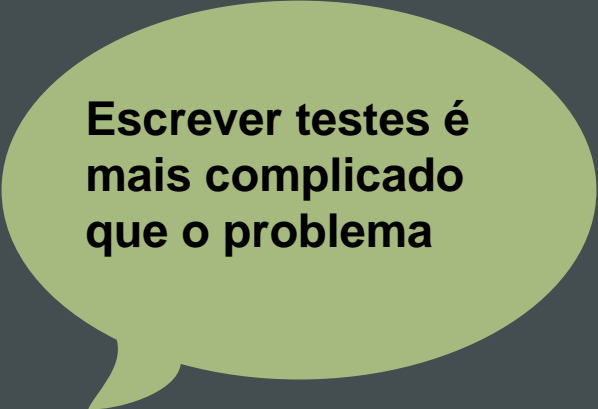
Prós e Cons

- Não é TDD
- Funciona mas beira Cowboyismo
- Cobertura só sobre o código mais frágil
- Lembra teste de regressão

Por que lembra um teste de regressão?

Porque é.

São testes de regressão para você mesmo.



**Escrever testes é
mais complicado
que o problema**

Longo sim, complicado não

Especialmente longo para testes funcionais

django_test_utils, o último bastião dos preguiçosos

django-test-utils

```
$ pip install django-test-utils
```

settings.py

```
INSTALLED_APPS = (  
    ...  
    'south', # migracoes  
    'django_nose', # depois do south  
    'test_extensions', # depois do south  
    'test_utils', # depois do south  
    ...  
)
```

Você começa o servidor

```
$ ./manage.py testmaker -a forum
```

Cria testes para você

```
Handling app 'forum'
Logging tests to foobar/forum/tests/forum_testmaker.py
Appending to current log file
Inserting TestMaker logging server...
Validating models...
0 errors found

Django version 1.2.1, using settings 'foobar.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Quando você termina

```
$ cd forum/tests  
$ ls forum*  
forum_testdata.serialized  
forum_testmaker.py
```


Testes gerados

```
def test_forum_127958317459(self):  
    r = self.client.get('/forum/', {})  
    self.assertEqual(r.status_code, 200)  
    self.assertEqual(  
        unicode(r.context["paginator"]), u"None")  
    self.assertEqual(  
        unicode(r.context["object_list"]),  
        u"<Topico: Topico object>, <Topico: Topico object>")  
    .....
```



**Eu conserto os
testes depois**

PFFFFFFFFFFFFFFF!

**TDD não é difícil.
Difícil é não fazer
quando voce
acostuma**

Créditos

<http://www.flickr.com/photos/blue-moose/3528603529>

Dúvidas?

Agradecimentos

<http://associacao.python.org.br/>

Nos vemos na PythonBrasil[6] em
Curitiba

Outubro 21 a 23

Referências

```
http://code.google.com/p/python-nose/  
http://github.com/jbalogh/django-nose  
http://github.com/garethr/django-test-extensions  
github.com/ericholscher/django-test-utils  
github.com/ctb/pony-build
```

Tdd em django sem desculpas
@fractal
petrich@gmail.com
creative commons (by) (sa)