

Testando Aplicações Django

Como? Quando? Onde?

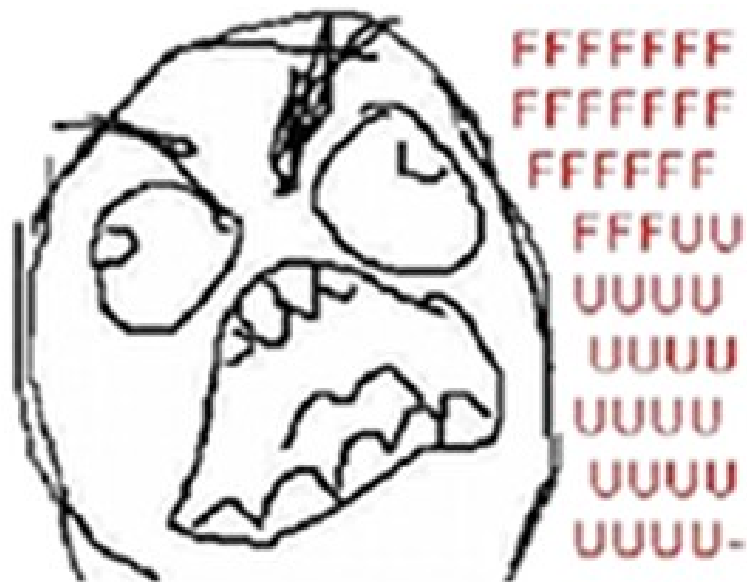
Bernardo Fontes
@bbfontes

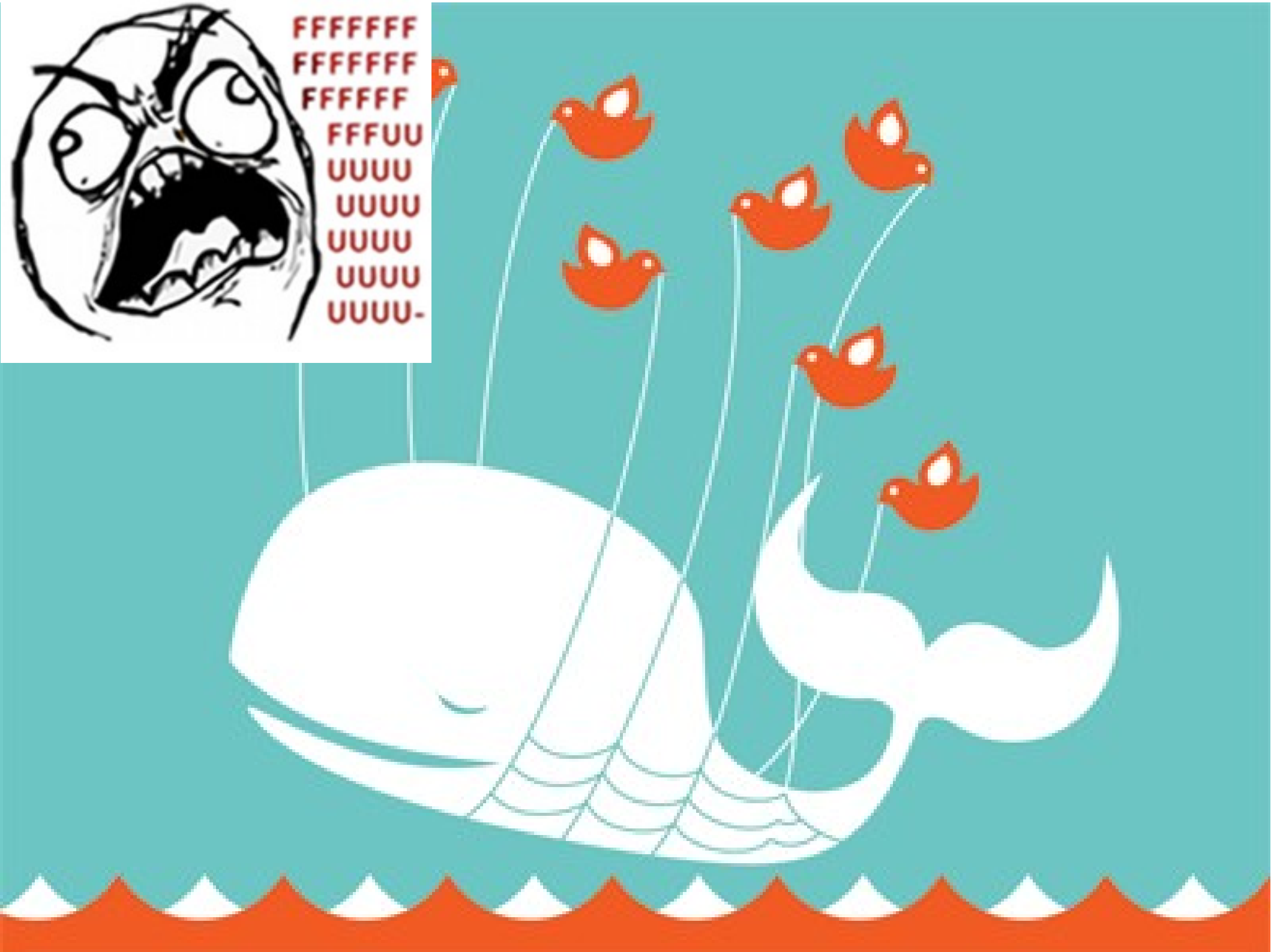


@bbfontes - bernardofontes.net - falecomigo@bernardofontes.net



Por que testar?





Windows



An exception 06 has occurred at 0028:C11B3ADC in VxD DISKTSU(03) + 00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) + 00000000. It may be possible to continue normally.

- * Press any key to attempt to continue.
- * Press CTRL+ALT+RESET to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue

Testes não são garantias

Testes não são garantias,
mas são excelentes
indicadores

Por que falar
sobre **testes**?

Só escrever testes
não basta

Teste também é código!



I'm *watching*!

Código deve ser
fácil de ler.

Código deve ser
fácil de ler.

Testes também!

Código deve ser
fácil de entender.

Código deve ser
fácil de entender.

Testes também!

Código deve ser
fácil de manter.

Código deve ser
fácil de manter.

Testes também!

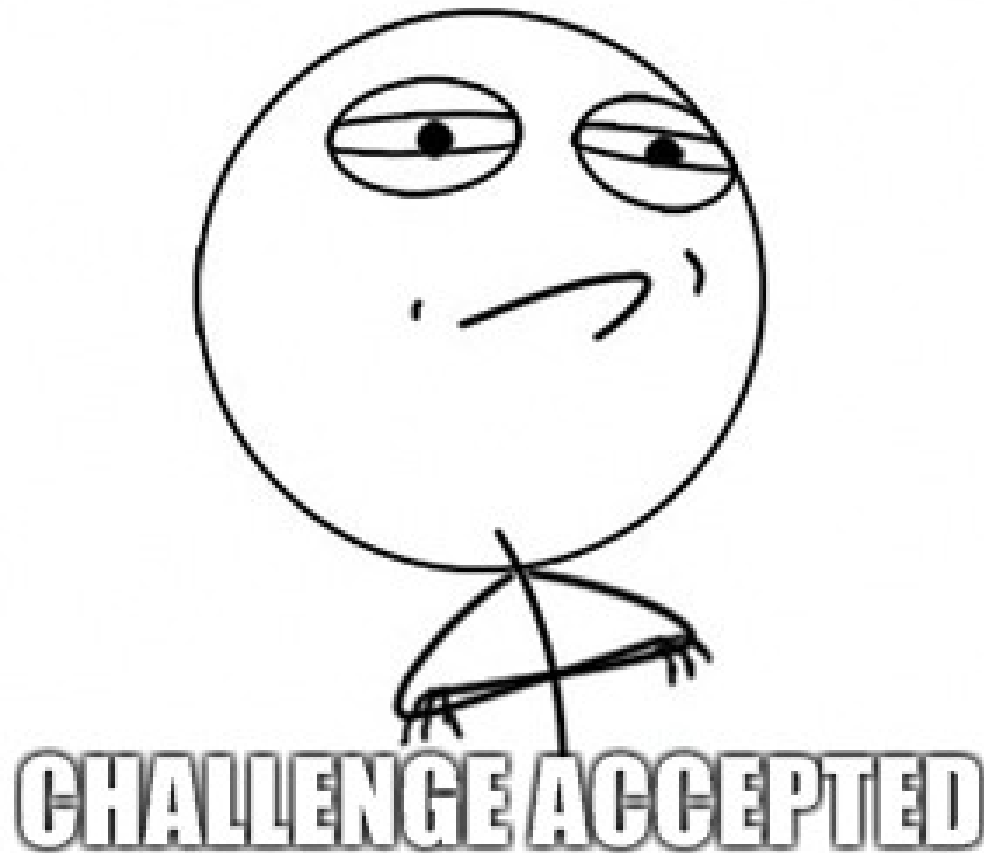
#NOT

```
256 def test_register_view(self):
257     """
258     Test that after registration a key cannot be reused.
259
260     """
261     # The first use of the key to register a new user works.
262     registration_data = self.sample_registration_data.copy()
263     self.sample_key.recipient_email = 'recipient@email.com'
264     self.sample_key.save()
265
266     # Make sure the form has a proper hidden invitation_key field
267     response = self.client.get(reverse('invitation:registration_register'),
268                                 {'invitation_key': registration_data['invitation_key']})
269     self.assertEqual(response.status_code, 200)
270     self.assertEqual(response.context['form'].initial.get('invitation_key'), registration_data['invitation_key'])
271     self.assertEqual(response.context['form'].initial.get('email'), self.sample_key.recipient_email)
272     self.assertEqual(response.context['form']['invitation_key'].is_hidden, True)
273
274     # The first use of the key to register a new user works.
275
276     response = self.client.post(reverse('invitation:registration_register'),
277                                 data=registration_data)
278     self.assertRedirects(response, reverse('invitation:registration_complete'))
279     user = User.objects.get(username='newuser')
280     key = InvitationKey.objects.get_key(self.sample_key.key)
281     self.assertEqual(user, key.registrant)
282
283     # Trying to reuse the same key then fails.
284     registration_data['username'] = 'even_newer_user'
285     response = self.client.post(reverse('invitation:registration_register'),
286                                 data=registration_data)
287     self.assertEqual(response.status_code, 200)
288     self.assertTemplateUsed(response,
289                             'invitation/wrong_invitation_key.html')
```



E dá para escrever bons
testes no Django?

E dá para escrever bons
testes no Django?




Recursos do Django

- Views
- Templates
- Forms
- Models
- Signals
- Template tags
- Template filters
- Settings
- ...

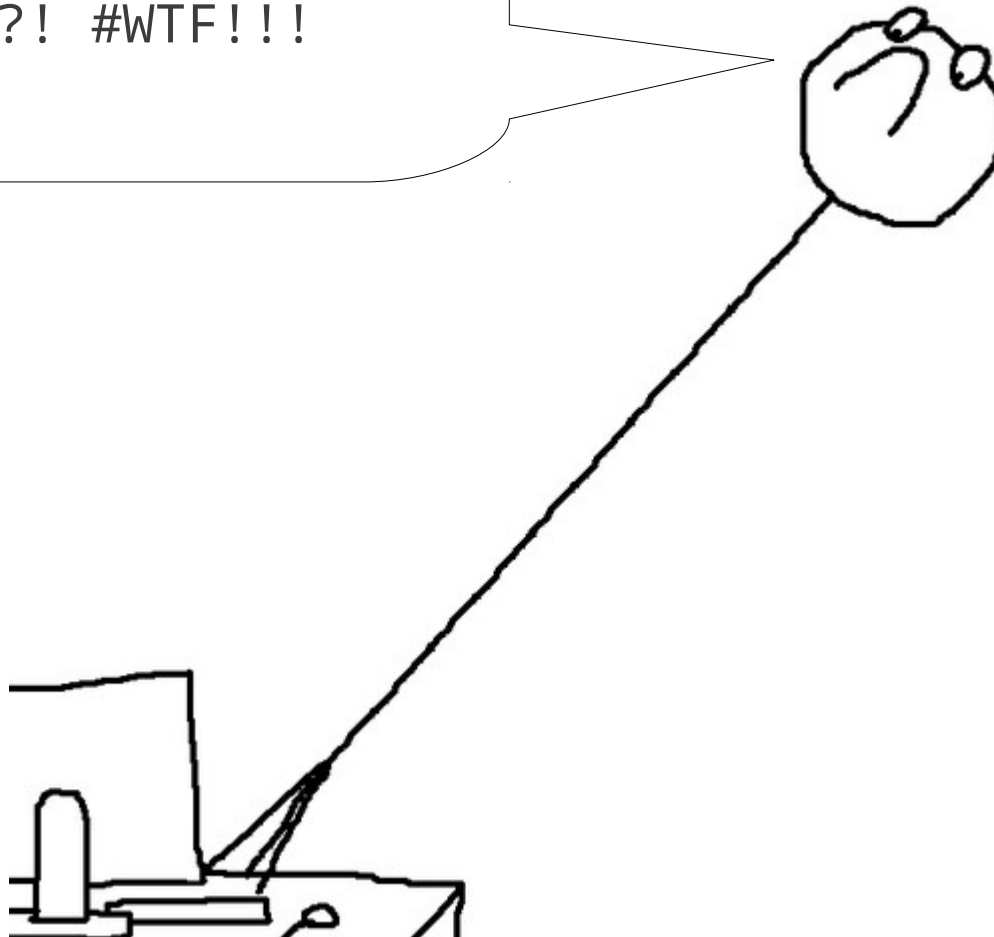
Como começar a testar?

python manage.py startapp core

```
.. (up a dir)
<tualenvs/pythonbrasil/project/
| ~core/
| | -__init__.py
| | -models.py
| | -tests.py ←
| ` -views.py
| -__init__.py
| -manage.py
| -settings.py
` -urls.py
```



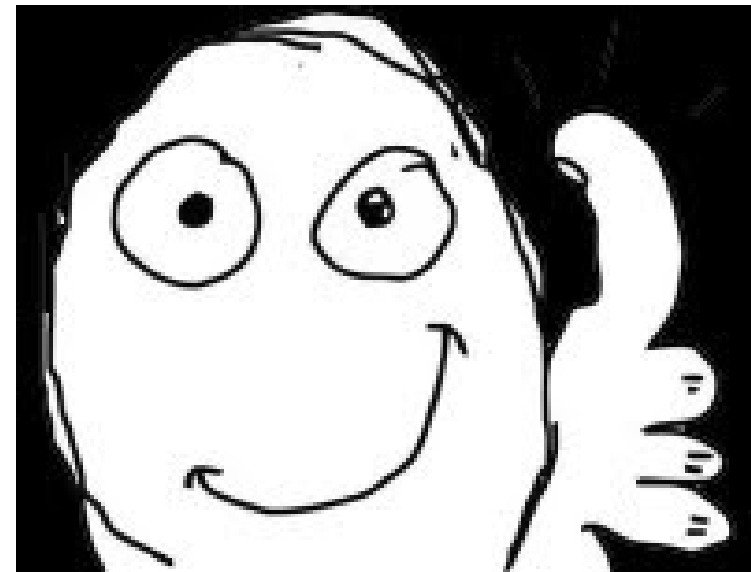
TUDO NUM ARQUIVO .PY
SÓ!?!?! #WTF!!!



Organize seus testes

Test as Packages

```
.. (up a dir)
<tualenvs/pythonbrasil/project/
| ~core/
| | ~tests/
| | `--__init__.py
| | -__init__.py
| | -models.py
| | `--views.py
| -__init__.py
| -manage.py
| -settings.py
| `--urls.py
```



Test as Packages

```
| ~core/
```

```
| | ~tests/
```

```
| | | -__init__.py
```

```
| | | -test_forms.py
```

```
| | | -test_models.py
```

```
| | | -test_views.py
```

```
| | -__init__.py
```

```
| | -forms.py
```

```
| | -models.py
```

```
| | -views.py
```

```
| -__init__.py
```

```
| -manage.py*
```

```
| -settings.py
```

```
| -urls.py
```

Test as Packages

```
| ~core/
```

```
| | ~tests/
```

```
| | | -__init__.py
```

```
| | | -test_forms.py
```

```
| | | -test_models.py
```

```
| | | -test_views.py
```

```
| | -__init__.py
```

```
| | -forms.py
```

```
| | -models.py
```

```
| | -views.py
```

```
| -__init__.py
```

```
| -manage.py*
```

```
| -settings.py
```

```
`-urls.py
```

__init__.py:

```
1 from test_forms import *
```

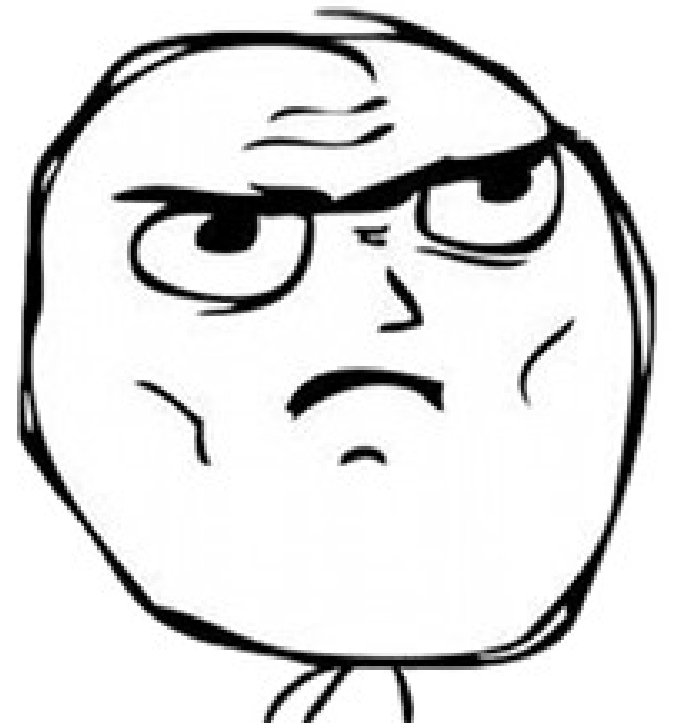
```
2 from test_models import *
```

```
3 from test_views import *
```

Explícito é melhor
que implícito e
evita ambiguidades

Testando o tripé do Django

- Views
- Forms
- Models



Views – 0 que testar?

- Em 99% dos casos:
 - Status code da resposta
 - Template utilizado
 - Contexto da resposta
 - Persistência no banco de dados
- Em alguns outros casos:
 - Envio de e-mails
 - Conteúdo da resposta
 - Disparo de sinais
 - Conexões com APIs externas

Views – Como testar?

Django TestClient:

- `get(path, data={}, follow=False, **extra)`
- `post(path, data={}, content_type=MULTIPART_CONTENT, follow=False, **extra)`
- `put(path, data={}, content_type=MULTIPART_CONTENT, follow=False, **extra)`
- `delete(path, follow=False, **extra)`
- ...
- `login(**credentials)`
- `logout()`

Testando uma view básica

```
15 class SubscriptionCreateViewTest(TestCase):
16
17     def test_show_form_on_get(self):
18         response = self.client.get(reverse('subscription:form'))
19         self.assertEqual(200, response.status_code)
20         self.assertTemplateUsed(response, 'subscription/subscription_form.html')
21
22     def test_has_form_on_context(self):
23         response = self.client.get(reverse('subscription:form'))
24         self.assertIsInstance(response.context['form'], SubscriptionForm)
25
26     def test_show_form_with_errors(self):
27         response = self.client.post(reverse('subscription:form'), data={})
28         self.assertEqual(200, response.status_code)
29         self.assertTemplateUsed(response, 'subscription/subscription_form.html')
30         self.assertIsInstance(response.context['form'], SubscriptionForm)
31         self.assertTrue(response.context['form'].errors)
32
33     def test_save_new_subscription(self):
34         self.assertEqual(Subscription.objects.count(), 0)
35         response = self.client.post('subscription:form', self.valid_post_dict)
36         self.assertEqual(Subscription.objects.count(), 1)
37
38     def test_redirects_after_save(self):
39         response = self.client.post('subscription:form', self.valid_post_dict)
40         obj = Subscription.objects.get(pk=1)
41         self.assertRedirects(response, reverse('subscription:checkout', args=[obj.hash]))
```

Mais complexidade com email

```
47 def test_send_welcome_email_for_new_subscription(self):
48     self.assertEqual(len(mail.outbox), 0)
49     response = self.client.post('subscription:form', self.valid_post_dict)
50     self.assertEqual(len(mail.outbox), 1)
51
52 def test_send_welcome_email_construction(self):
53     response = self.client.post('subscription:form', self.valid_post_dict)
54     obj = Subscription.objects.get(pk=1)
55     email = mail.outbox[0]
56     self.assertRecipients(email, [obj.email, settings.DEFAULT_FROM_EMAIL])
57     self.assertEqual(email.from_email, 'inscricao@pythonbrasil.com.br')
```

Mais complexidade com conteúdo da resposta

```
5 class UserInfoApiTest(TestCase):
6
7     def test_correct_content_type(self):
8         response = self.client.post('core:get_user', data={'user_id':1})
9
10        self.assertEqual(200, response.status_code)
11        self.assertEqual('application/json', response['content-type'])
12
13    def test_correct_content_data(self):
14        response = self.client.post('core:get_user', data={'user_id':1})
15        returned_json = json.loads(response.content)
16        user = User.objects.get(id=1)
17
18        self.assertEqual(returned_json['id'], user.id)
19        self.assertEqual(returned_json['full_name'], user.full_name())
```

Testando uma view do admin

```
66 def test_send_charging_email(self):
67     self.assertEqual(len(mail.outbox), 0)
68     self.client.login(username='admin', password='admin')
69     response = self.client.get('admin:send_charging_mail', args=[1])
70
71     self.assertEqual(len(mail.outbox), 1)
72
73 def test_404_when_subscription_doesnt_exist(self):
74     self.client.login(username='admin', password='admin')
75     response = self.client.get(reverse('admin:send_charging_mail', args=[100]))
76
77     self.assertEqual(response.status_code, 404)
```

Forms – O que testar?

- Validação do formulário
 - Geral
 - Por campo
- Comportamento dos campos
 - Ex: atributo choice do `ModelChoiceField`
- Comportamento do domínio do form
 - Ex: método `save` sobreescrito

Forms – Como testar?

- Instanciar um objeto da classe do form com um dicionário de strings com os dados sendo passado por parâmetro

Forms – Code Snippet

```
class SubscriptionFormTest(TestCase):

    def test_hash_field_not_present(self):
        # model form excluded field
        form = SubscriptionForm()
        self.assertFalse(form.fields.get('hash'))

    def test_all_required_form_fields(self):
        form = SubscriptionForm({
            'name': '',
            'cpf': '',
            'email': '',
        })
        form.is_valid()
        self.assertIn('name', form.errors)
        self.assertIn('cpf', form.errors)
        self.assertIn('email', form.errors)

    def test_cpf_must_have_11_digits(self):
        form = SubscriptionForm({
            'name': 'Bernardo Fontes',
            'cpf': '123456789',
            'email': 'falecomigo@bernardofontes.net',
        })
        form.is_valid()
        self.assertIn('cpf', form.errors)
```

Models – O que testar?

- Validação do modelo
- Métodos adicionados ao modelo
- Estados da queryset
- Cuidado para não testar o Django

Models – Como testar?

- Instanciação de objetos daquele modelo normalmente e asserções através da API daquele objeto;

Models - Snippet

```
7 class SubscriptionModelTest(TestCase):
8
9     def setUp(self):
10         self.data = {
11             'name': 'Bernardo Fontes',
12             'email': 'falecomigo@bernarfontes.net',
13             'cpf': '1111111111',
14         }
15
16     def test_create_new_subscription(self):
17         s = Subscription.objects.create(**self.data)
18         self.assertEqual(s.id, 1)
19
20     def test_subscription_hash_on_save(self):
21         s = Subscription(**self.data)
22         self.assertFalse(s.hash)
23         s.save()
24         self.assertTrue(s.hash)
25
26     def test_doesnt_change_subscription_hash_if_set(self):
27         s = Subscription.objects.create(**self.data)
28         hash_value = s.hash
29         self.assertTrue(hash_value)
30         s.email = 'bernardo@decode.com.br'
31         s.save()
32         self.assertEqual(hash_value, s.hash)
33
34     def test_non_unique_fields(self):
35         Subscription.objects.create(**self.data)
36         Subscription.objects.create(**self.data)
37         self.assertEqual(Subscription.objects.count(), 2)
```

Sagacidades com testes!

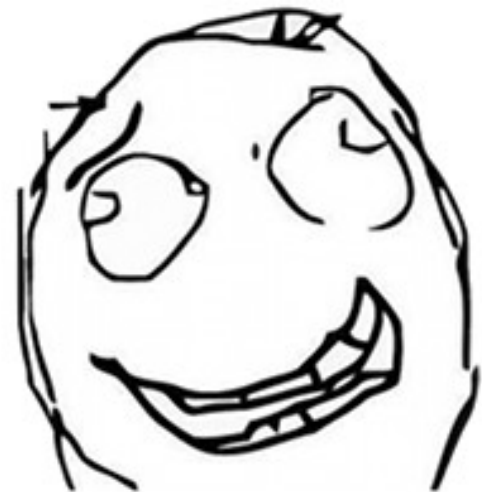
Estude TODO o pacote `django.test`

<https://docs.djangoproject.com/en/dev/intro/tutorial05/>
<https://docs.djangoproject.com/en/dev/topics/testing/advanced/>



Sagacidades com testes!

Use os asserts que o Django já te fornece.



Sagacidades com testes!

Extenda o TestCase do Django adicionando novos métodos de testes.

Mas cuidado para não fazer
#tudojuntoemisturado!



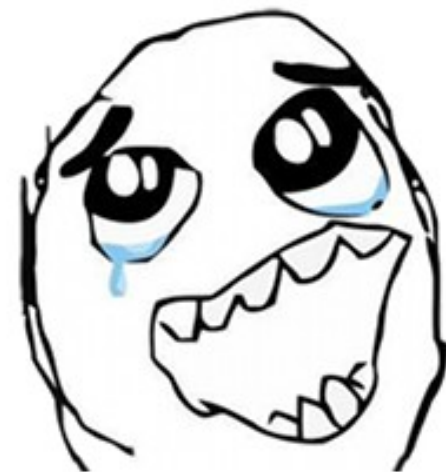
Sagacidades com testes!

```
43 class TestCase(test.TestCase):
44     client_class = ReverserClient
45
46     def assertQuerySetEqual(self, qs1, qs2, *args, **kwargs):
47         self.assertEqual(list(qs1), list(qs2), *args, **kwargs)
48
49     def assertRecipients(self, email, recipients):
50         email_recipients = email.recipients()
51         self.assertEqual(len(email_recipients), len(recipients))
52         for r in recipients:
53             self.assertIn(r, email_recipients)
```



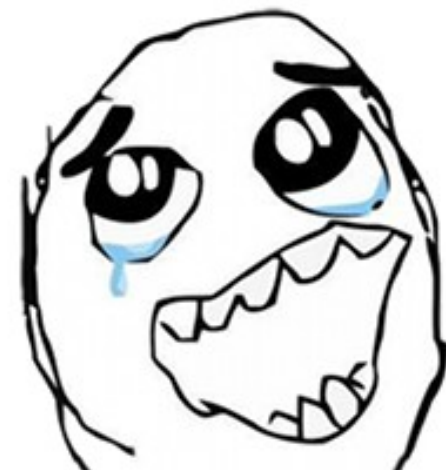
Sagacidades com testes!

Utilize o mock para simular comportamentos externos.



Sagacidades com testes!

```
@patch.object(PaymentApi, 'request_for_token', Mock(return_value='test_token'))
def should_set_token_on_order(self):
    response = self.client.get(self.url)
    order = PurchaseOrder.objects.get(id=self.order.id)
    self.assertEqual('test_token', order.payment_token)
```



Sagacidades com testes!

Pré-popule seu banco utilizando o
`model_mommy`.

http://github.com/vandersonmota/model_mommy/



Sagacidades para escrever testes!

```
54 class CheckoutViewTest(TestCase):
55
56     def setUp(self):
57         user = mommy.make_one(User)
58         self.signature = mommy.make_one(Signature, user=user)
59
60     def test_show_checkout_page_on_get(self):
61         response = self.client.get('signature:checkout', args=[self.signature.hash])
62         self.assertTemplateUsed(response, 'signature/signature_checkout.html')
```



Sagacidades com testes!

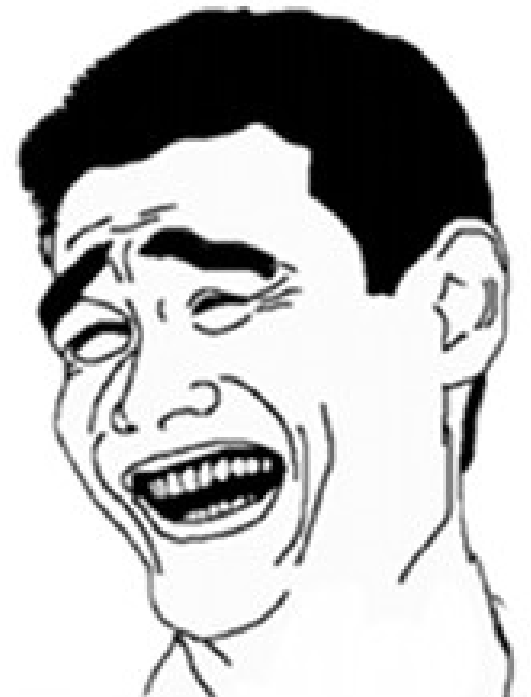
Substitua o TestRunner
do Django pelo Nose.



<http://code.google.com/p/python/python-nose/>

Ainda faltou testarmos

- Signals
- Template filters
- Template tags
- Settings
- Management commands
- ModelAdmin
- Forms widgets
- ...



Obrigado!



Perguntas?

