04 o que não te contaram sobre django

Settings

Existem algumas fraquezas em usar um conjunto de settings globais para todos os ambientes.

3 hacks para melhorar seu settings.py

`PROJECT_PATH`

`settings_local.py`

`auto debug`

PROJECT_PATH

A criação dinamica de um PROJECT_PATH facilita a execução em multiplos ambientes

```
import os
import sys

PROJECT_PATH = os.path.abspath(os.path.split(__file__)[0])
...
MEDIA_ROOT = os.path.join(PROJECT_PATH,'media')
```

settings_local.py

se existir `settings_local.py` executa ela

```
# ultima coisa no settings
try:
    execfile(PROJECT_PATH+'/settings_local.py')
    #print 'Usando configuracao LOCAL'
except IOError:
    #print 'Usando configuracao PADRAO'
    pass
```

auto debug

O menos aconselhavel das 3 modificações

```
import socket

# Set DEBUG = True if on the production server
if socket.gethostname() == 'your.domain.com':
    DEBUG = False
else:
    DEBUG = True

TEMPLATE_DEBUG = DEBUG
```

Bonus hack

no `settings.py`

```
TEMPLATE_TAGS = (
        'projeto.foo.templatetags.foobar',
        'projeto.bar.templatetags.foobar',
)
```

autoload template_tags

no __init__.py

```
from django.conf import settings
from django.template import add_to_builtins
#import django.template.loader

try:
    for lib in settings.TEMPLATE_TAGS:
        add_to_builtins(lib)
except AttributeError:
    pass
```

Costumizando o Admin

criar o diretorio de templates especial pro admin

```
mkdir -p templates/admin

cd templates/admin
```

base_site.html

```
{% extends "admin/base_index.html" %}

{% block title %}{{ title }} | Admin da pizzaria {% endblock %}

{% block branding %}
<h1 id="site-name" > Pizzaria</h1>
{% endblock %}
```

existe ainda um {% block extrastyle %} para colocar styles de css

OBS: incluir tambem o <style> e </style>

Applicações para conhecer

`django-config` Gerencia varias configuracoes

`django-debug-toolbar` Fantastico!

`haystack` Facilita search

`sphinx` Grande ferramenta de documentção

`celery` Controle de tarefas assincronamente

`fabric` Deploy e gerenciamento remoto

`gnucorn` Servidor WSGI

`varnish` Solucao para cache

Decoradores

Recebe uma função como parametro e retorna uma função

Exemplo

```
def decorador(func):
    func.tipo = "decorada"
    return func

@decorador
def foo(valor):
    print valor
# python antigo
# foo = decorador(foo)
```

usando

```
>>> foo("oi")
oi

>>> foo.tipo
'decorada'
```

funcoes dentro de funcoes

```
def decorador(func):
    def nova_f(*args):
        print "iniciando", func.__name__
        func(*args)
        print "terminando"
    return nova_f
```

executando

```
>>> foo("oi")
iniciando  foo
oi
terminando
>>> print foo.__name__
'nova_f'
```

decoradores com parametros

```
@decorador("legal")
def foo(valor):
    print valor
```

Fabrica de decoradores

```
def decorador(tipo):
    def fabrica(func):
        def nova_f(*args):
            print "tipo ", tipo
            print "iniciando ", func.__name__
            func(*args)
            print "terminando"
        return nova_f
    return fabrica
```

python < 2.4

```
foo = decorador("legal")(foo)
```

decoradores como Classes

```
class decorador(object):
    def __init__(self, func):
        self.f = func

    def __call__(self,*args):
        print "iniciando "
        self.f(*args)
        print "terminando"
```

Decoradores importantes

```python
# do python
@staticmethod
@property

#do django
@login_required
def my_view(request):
    ...
```

```python
from django.contrib.auth.decorators import login_required
```

Signals

Sinais não são o que voce pensa

Eles são:

Sincronos

Rolam na mesma thread

Exemplo de uso

modificar um valor antes de salvar

```
from django.db.models import signals

class Foo(models.Model):
    validade = models.DateTimeField()

def marca_val(sender,instance,**kwargs):
    if not instance.validade:
        instance.validade = "2012-01-01"
signals.pre_save.connect(marca_val, sender=Foo)
```

No shell

```
>>>from larari.models import Foo
>>>a = Foo()
>>>a.validade

>>> a.save()
>>> a.validade
'2012-01-01'
#ou datetime.datetime(2012, 1, 1, 0, 0)
```

DB Signals

pre_save, post_save, pre_delete, post_delete Requer o parametro sender

enviam sender, instance e outros(post_save manda o created)

m2m_changed

alem destes manda ainda action veja a documentação

http://docs.djangoproject.com/en/dev/ref/signals/

Signals de gerencia

```
post_syncdb

request_started

request_finished

got_request_exception

connection_created
```

Signals que so rolam em testes

`template_rendered`

Seus Signals

```
from django.dispatch import Signal

compra_pronta = Signal(providing_args=["c_id"])

class Compra():
    def finalizar_compra(self):
        ...
        compra_pronta.send(sender=self,c_id=self.id)
```

Ouvindo seus signals

```
def compra_callback(sender,c_id,**kwargs):
    pass

compra_pronta.connect(compra_callback)
```

cache

```
CACHE_BACKEND = 'memcached://127.0.0.1:11211/'

# tem que fazer no shell antes:
# ./manage.py createcachetable [cache_table_name]
CACHE_BACKEND = 'db://my_cache_table'


CACHE_BACKEND = 'file:///var/tmp/django_cache'

CACHE_BACKEND = 'locmem://'

CACHE_BACKEND = 'dummy://' # NAO CACHEIA so pra testes
```

middlewares

```
MIDDLEWARE_CLASSES = (
    # tem que ser o primeiro
    'django.middleware.cache.UpdateCacheMiddleware',
    ...
    # tem que ser o ultimo
    'django.middleware.cache.FetchFromCacheMiddleware',
)
CACHE_MIDDLEWARE_SECONDS  = 90 #segundos
CACHE_MIDDLEWARE_KEY_PREFIX = "" # somente multiplos sites
```

Tipos de cache

por view

template fragment cache

API de baixo nivel

Tipos de cache

por view

template fragment cache

API de médio nivel

por view

```
from django.views.decorators.cache import cache_page

@cache_page( 90 ) #segundos
def foo(request):
    ...
```

ou direto na urls.py

urls.py

```
from django.views.decorators.cache import cache_page

urlpatterns = ('',
    (r'^foo/$', cache_page(foo, 90)),
)
```

template fragment cache

legal demais e simples demais

```
{% load cache %}
{% cache 500 barra_menu %}
    .. barra_menu ..
{% endcache %}
```

Mas

E para usuários logados?

Valores dinamicos?

Cache por usuário

```
{% load cache %}
{% cache 500 barra request.user.username %}
    ... barra do usuario ...
{% endcache %}
```

API de médio nivel

```
>>> from django.core.cache import cache
>>> timeout = 5
>>> cache.set("chave","objeto",timeout)
>>> cache.get("chave")
"objeto"
>>> cache.get("chave")
None
```

Mais metodos

```
cache.clear()

cache.add(chave,valor)
cache.get(chave,valor_padrao)
cache.delete(chave)
# django 1.2
cache.set_many({chave1:val1, chave2:val2})
cache.get_many([chave1,chave2,chave3])
cache.delete_many([chave1,chave2,chave3])
```

cache para contadores

```
cache.set("contador", 1)
cache.incr("contador")

cache.incr("contador",5)

cache.decr("contador")
```

cache entre o django e o browser

```
from django.views.decorators.vary import vary_on_headers

@vary_on_headers('User-Agent')
def foo(request):
    ...
```

cookies

```
from django.views.decorators.vary import vary_on_cookie

@vary_on_cookie
def foo(request):
    ...
```

private

```
from django.views.decorators.cache import cache_control

@cache_control(private=True)
def foo(request):
    ...
```

ou forca cache

```
from django.views.decorators.cache import cache_control

@cache_control(must_revalidate=True, max_age=3600)
def foo(request):
    ...
```

sem cache

```
from django.views.decorators.cache import never_cache

@never_cache
def foo(request):
    ...
```

permisso permisso

```
if request.user.has_perm('aluno.delete_entrega'):
    pass
```

virutalenv

```
Criar um ambiente virtual limpo:
$ virtualenv --no-site-packages ambiente

Ativar o ambiente virtual:
$ source ambiente/bin/activate

Instalar django:
$ pip install django

Sair do ambiente virtual:
$ deactivate
```

```
import pdb pdb.set_trace()

import nose nose.tools.set_trace()
```