# Django

Framework web para perfeccionistas com prazos

Igor Sobreira www.igorsobreira.com

#### o que é um framework?

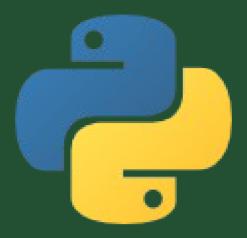
"é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido"

"Frameworks são projetados com a intenção de facilitar o desenvolvimento de software, habilitando designers e programadores a gastarem mais tempo determinando as exigências do software do que com detalhes de baixo nível do sistema."

# o que é django?

- framework web escrito em python
- desenvolvido no Lowrence Journal-World
- publicado em julho de 2005
- open-source (licensa BSD)

"Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design"



por que python?

- altíssimo nível
- multiparadigma (OO, funcional e procedural)
- tipagem dinâmica e forte
- coletor de lixo
- multiplataforma
- código limpo
- fácil de aprender

### "Produtividade e legibilidade"

# onde usar python?

- web
  - Django, TurboGears, Pylons. Zope, Plone, Grok...
- desktop
  - PyGTK, PyQT, WxPython, TkInter...
- mobile
  - PyS60 (Symbian), PyMaemo...

# por que usar django?

- feito em Python :-)
- MTV == MVC
- ORM (Mapeamento Objeto Relacional)
- Interface de Administração automática
- URLs limpas
- Boa documentação (+inglês)

#### separando as tarefas

MVC (Model – View – Controller)

MTV (Model – Template – View)

# organizando o código

um projeto possui uma ou mais aplicações

# DRY – Don't Repeat Yourself

# caminho de uma requisição

#### Request



#### urls.py

```
(r'^contato/$',
   'blog.contato'),
(r'^posts/$',
   'blog.posts'),
)
```



# Cliente

#### views.py

```
def posts(request):
    p=Post.objects.all()
    return render_to_response (
        'posts.html',
        {'posts':p}
    )
```



#### models.py

class Post(Model):
 title=CharField(max\_length=50)
 content=TextField()
 pub\_date=DateField()
 cat=ForeignKey(Category)
 autor=CharField(max\_length=50)

class Category(Model):
 name=CharField()



#### posts.html

```
{% extends "base.html" %}

{% block content %}

{% for p in posts %}

<h1> {{ p.title }} </h1>

 {{ p.content }} 
{% endfor %}
```



Response

#### models.py

from datetime import datetime from django.db import models

# models.py (2)

```
BEGIN;
CREATE TABLE "teste image" (
  "id" integer NOT NULL PRIMARY KEY,
  "image" varchar(100) NOT NULL,
  "description" varchar(50) NOT NULL,
  "pub date" datetime NOT NULL
COMMIT;
```

### models.py (3)

```
class Category(models.Model):
  name = models.CharField(max length=50)
class Post(models.Model):
  title = models.CharField(max length=50)
  content = models.TextField()
  pub date = models.DateTimeField(default=datetime.now)
  author = models.CharField(max length=50)
  category = models.ForeignKey(Category)
```

# models.py (4)

```
BEGIN;
CREATE TABLE "teste category" (
  "id" integer NOT NULL PRIMARY KEY,
  "name" varchar(50) NOT NULL
CREATE TABLE "teste post" (
  "id" integer NOT NULL PRIMARY KEY,
  "title" varchar(50) NOT NULL,
  "content" text NOT NULL,
  "pub date" datetime NOT NULL,
  "author" varchar(50) NOT NULL,
  "category id" integer NOT NULL REFERENCES
  "teste category" ("id")
COMMIT;
```

#### persistindo os dados

```
cat = Category(name = 'Django')
cat.save()
```

### persistindo os dados (2)

post.save()

#### acessando os dados

```
p = Post.objects.all()
```

[<Post: Post object>, <Post: Post object>]

#### acessando os dados (2)

```
p = Post.objects.filter(author = 'Igor')
```

#### acessando os dados (3)

```
p = Post.objects.get(id=2)
```

#### urls.py

from django.conf.urls.defaults import \*

```
urlpatters =patterns('myproject.myapp.views',
    (r'^lastest/$', 'last_posts'),
    (r'^post/(\d+)/$', 'post_details'),
    (r'^contact/$', 'contact'),
)
```

### views.py

from django.shortcuts import render\_to\_response from myapp.models import Post

#### templates

- hora dos web designers :-)
- simplesmente arquivos .html com "buracos" que serão preenchidos com o conteúdo passado pela view

```
<html>
<head>
 <title> Blog - {{ post.title }} </title>
</head>
<body>
 <h1> {{ post.title }} </h1>
  { { post.content } } 
 <small> Publicado por {{ post.author }} em
 {{ post.pub date }} </small>
</body>
</html>
```

#### templates (2)

- variáveis
  - {{ nome }}, {{ lista }}
- filters
  - {{ nome|safe }}
  - { { lista|unordered\_list } }
- tags
  - {% for nome in lista %} {{ nome }} {% endfor %}
  - {% if condicao %} <b> OK </b> {% endif %}

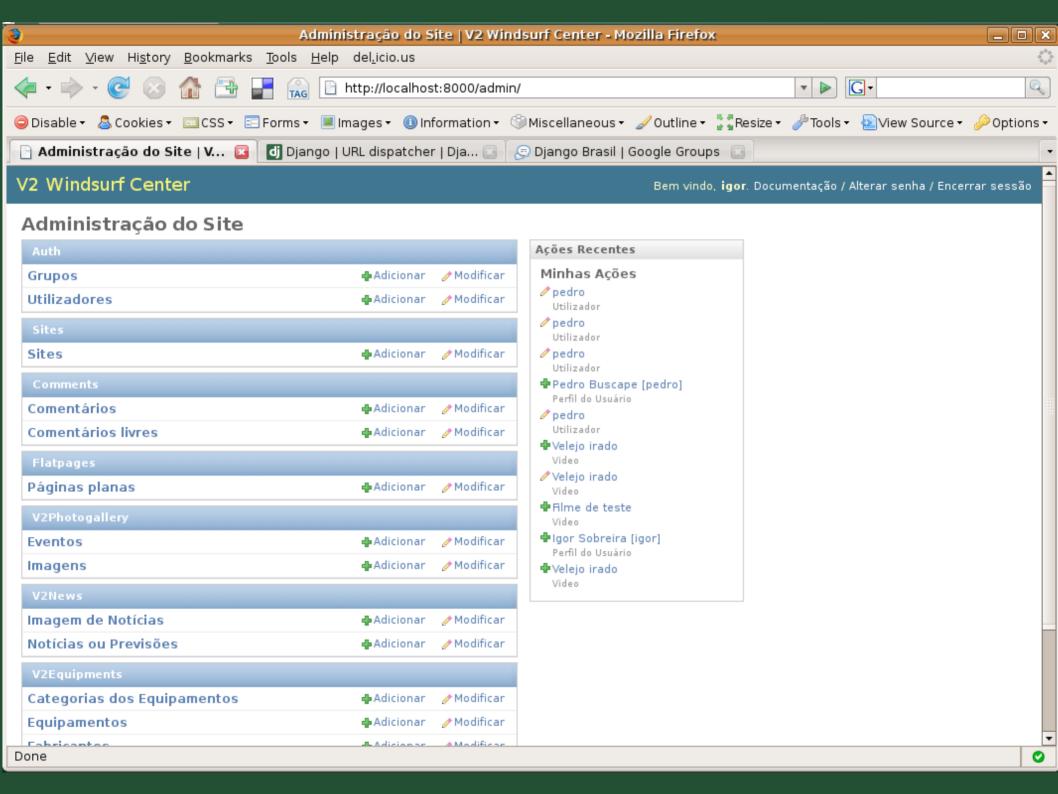
#### templates (3)

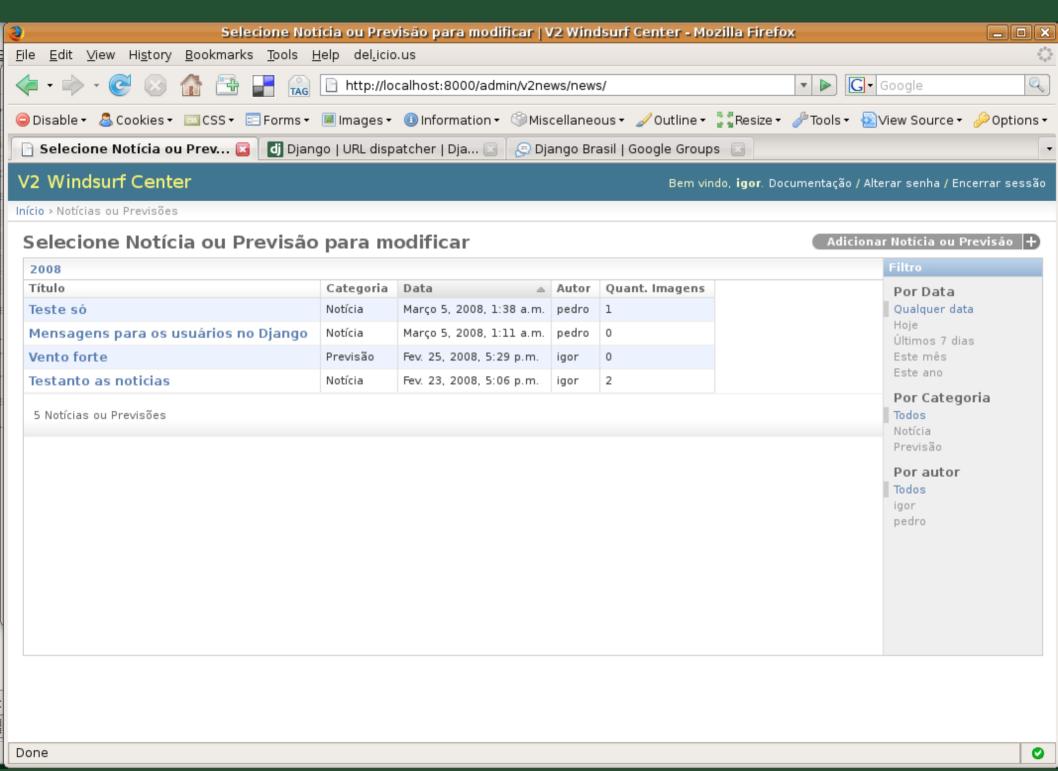
```
    {% for post in last_posts %}
    {| post|date:"d/m/Y h:i" }} -
        {{ post.title }} 
    {% endfor %}
```

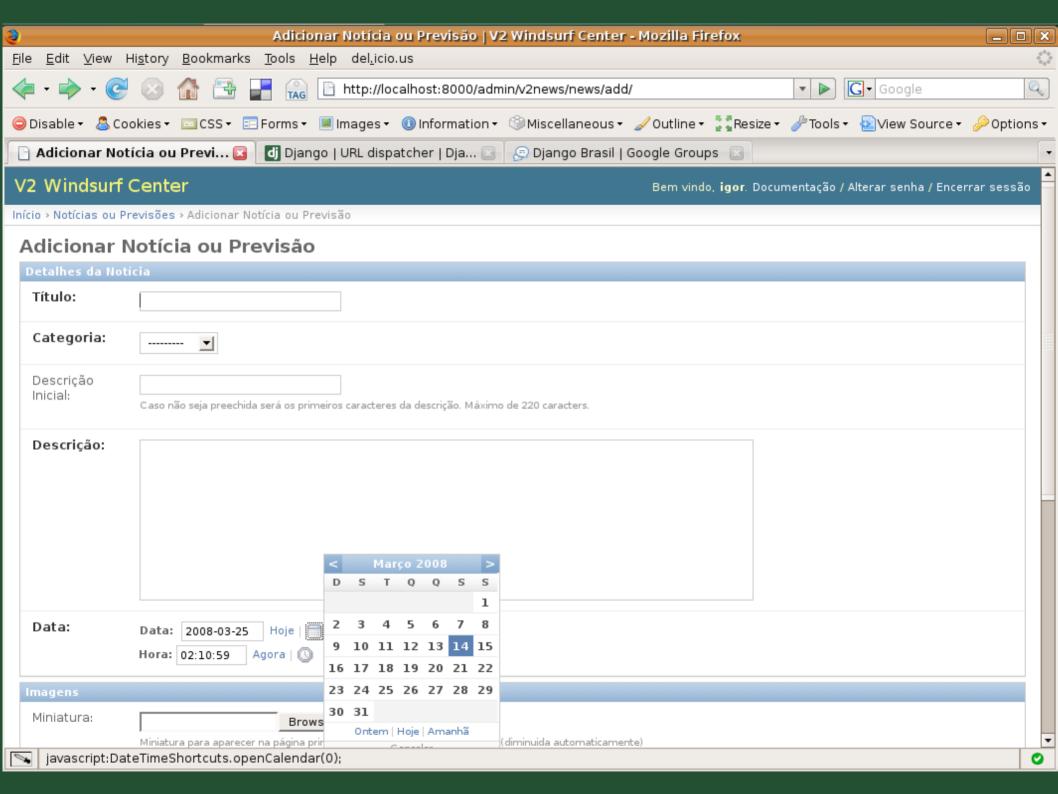
#### mais facilidades...

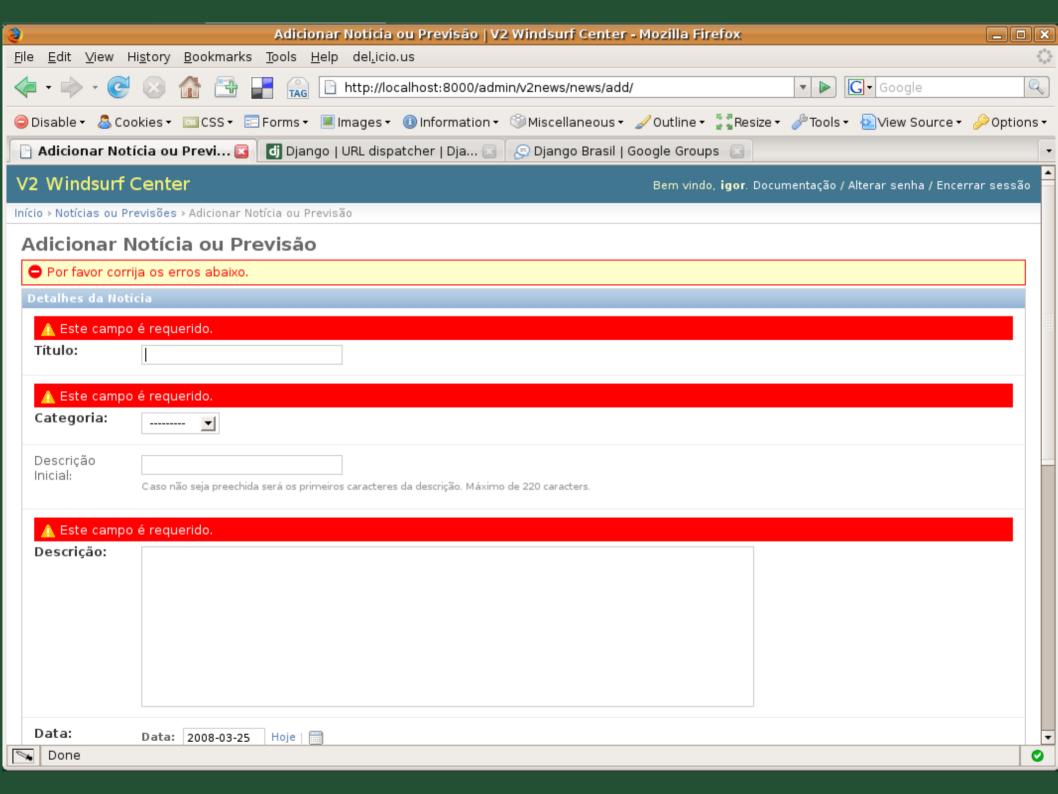
- formulários
- testes
- controle usuários (auth/auth)
- internacionalização (i18n)
- AJAX ?!?!
- feeds RSS / Atom
- cache

#### dando uma olhada no admin...









#### dúvidas

?

- http://www.djangoproject.com/
- http://www.djangobrasil.org/
- http://groups.google.com/group/django-brasil/
- http://djangobook.com/
- http://www.python.org
- http://www.pythonbrasil.com.br/
- irc.freenode.net (#django-br, #django)

#### Igor Sobreira

- www.igorsobreira.com
- http://djangopeople.net/igorsobreira
- G-Talk: v2.igor@gmail.com
- MSN: igor@v2windcenter.com
- IRC: igors

#### se der tempo...

#### formulários

from django import newforms as forms

# Formulários (contato.html)

```
<form action="." method="post">
{{ form.as_p }}
<input type="submit" value="Enviar" />
</form>
```

# Formulários (views.py)

```
def contato(request):
  if request.method == "POST":
     form = ContatoForm(request.POST)
     if form.is valid():
       # faz algo aqui...
        return HttpResponseRedirect("/sucesso/")
   else:
       form = ContatoForm()
   return render to response("contato.html",
                              {"form":form})
```