Desenvolvimento ágil com Python + Django

Dourados, 04 de Novembro de 2014

Gabriel Dieterich Cavalcante

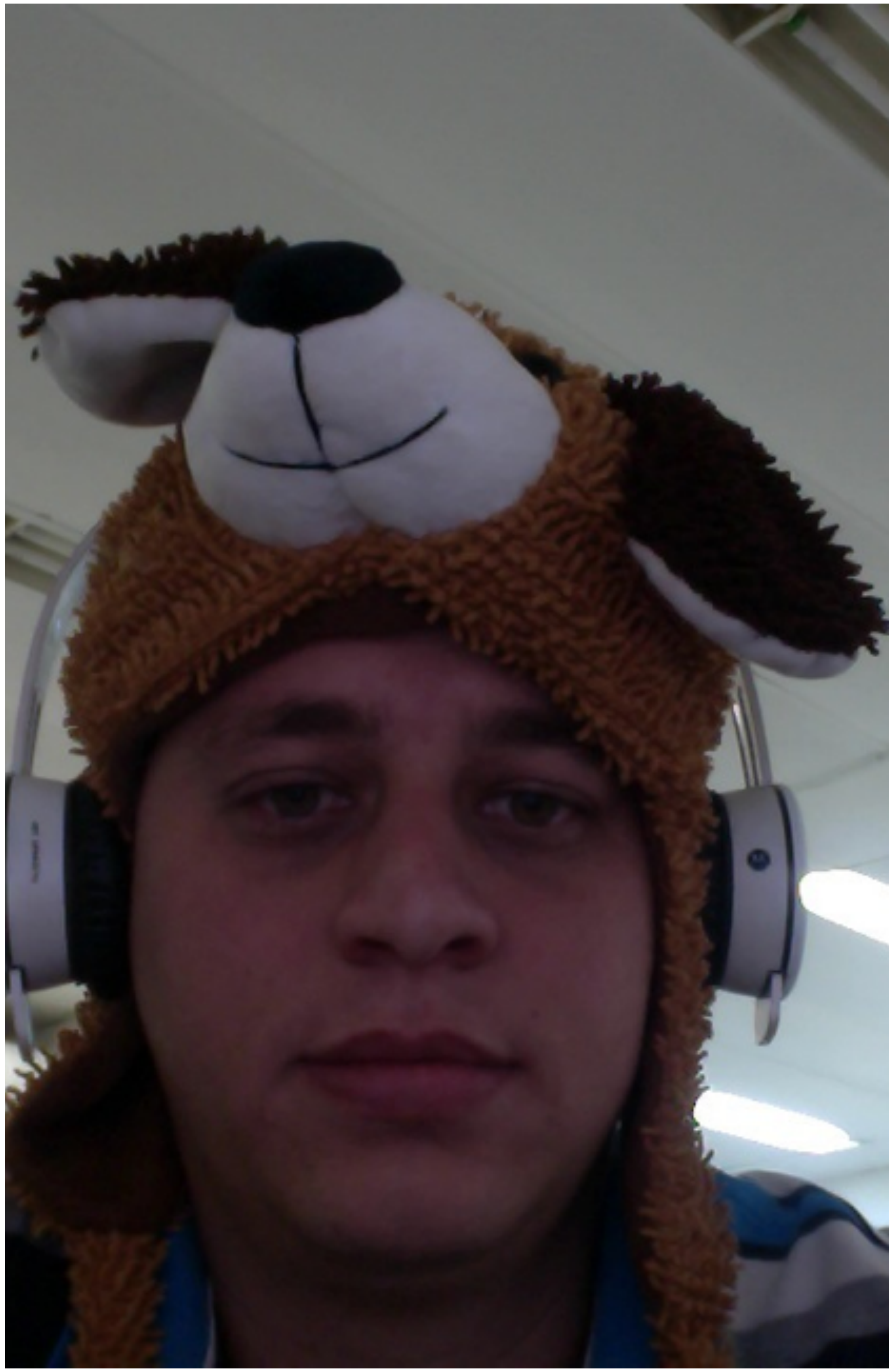
27 Anos aniversário em 14 dias, aceito presentes ;)

CC UEMS 2007
MSC UNICAMP 2010
PHD UNICAMP **working on it ;-D**

Google Summer of Code 2009 e 2010
1st Place Facebook Hackathon Brazil 2012
1st Place Facebook Hackathon Brazil 2013

2008 - 2010 Projeto Librix Linux (Unicamp)
2009 - * Consultoria Dev com Software Livre (C&A, Casas Bahia, Mag. Luiza, Unicamp, USP, Marinha etc)
2011 - * Partner at Elabora Consultoria

gabriel@elabsis.com / gdcavalcante@gmail.com
@escovabr
fb.com/gdcavalcante


Winter programming - 2013

Introdução e História

Django

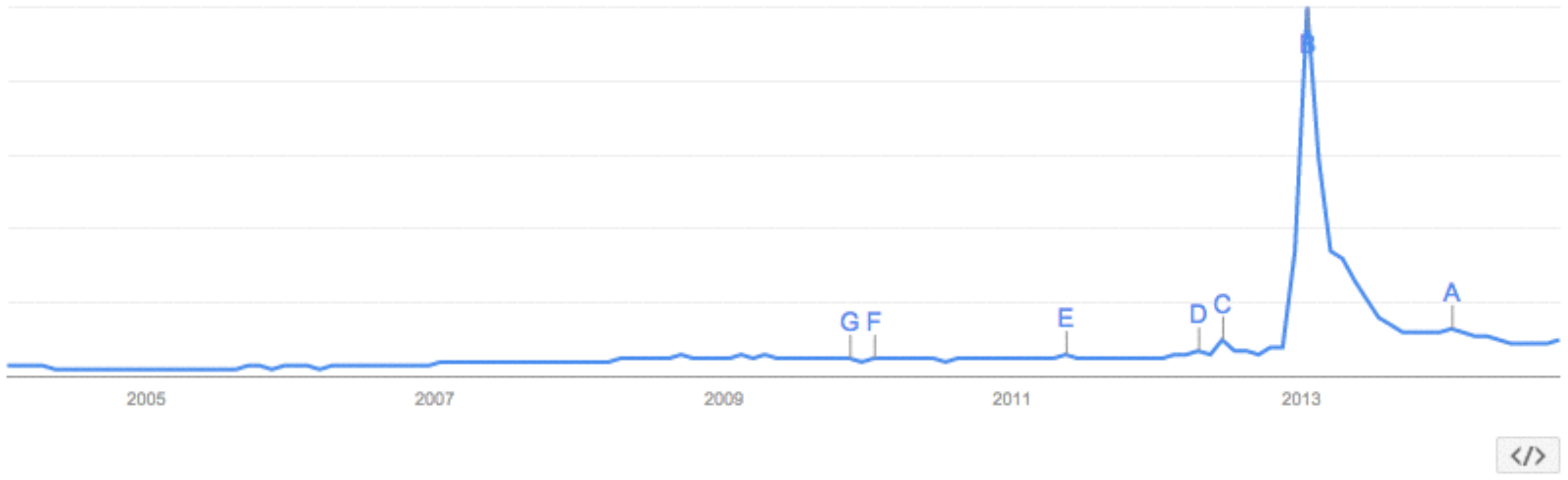
- A "web application framework"
  - Release publicy in 2005
  - basically a framework abstracted from web application for a newspaper in Kansas.
  - immediate popularity and lots of development activity
  - 1.0 release in 2008 (essentially modern Django)
  - 1.7 released earlier this year

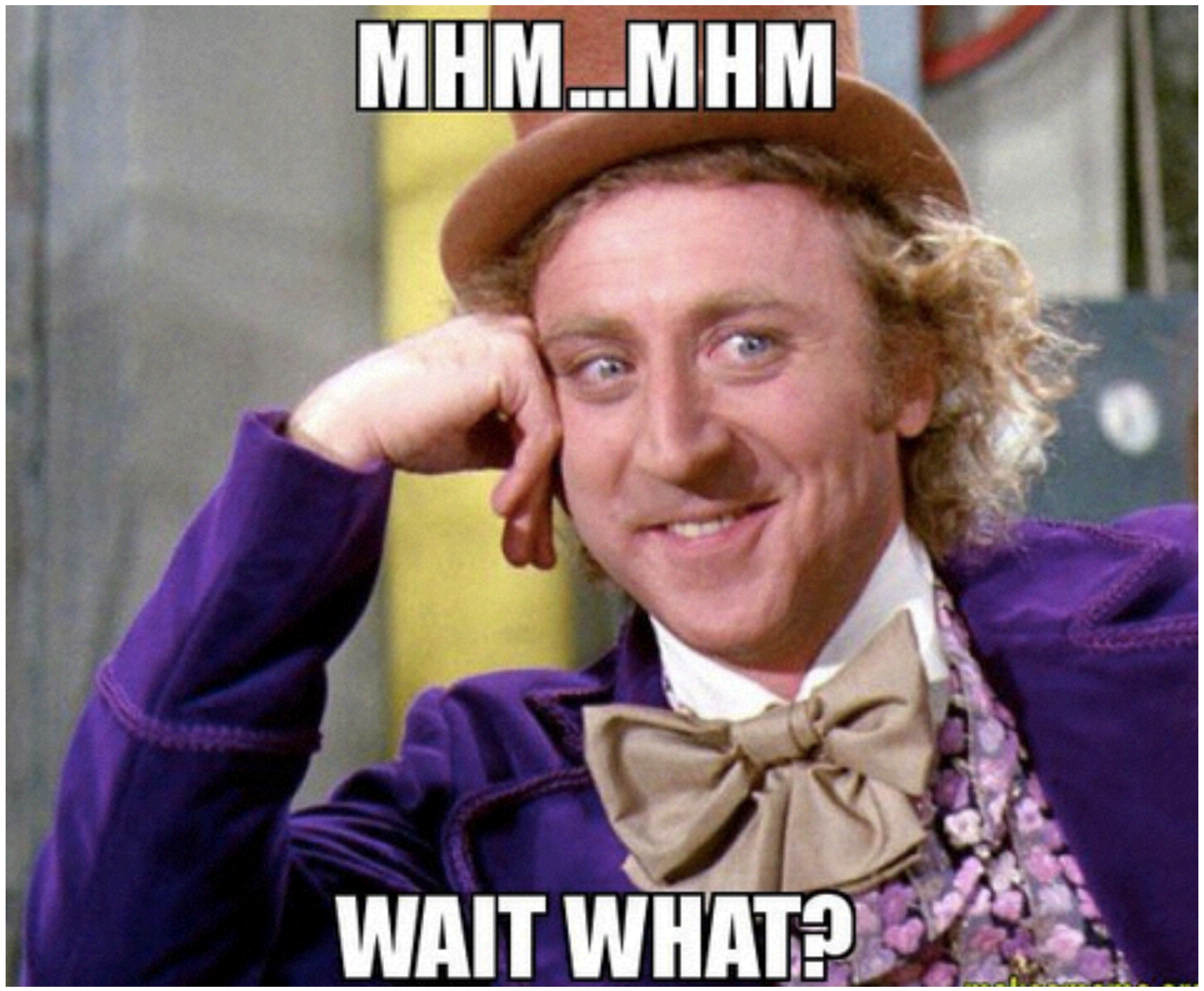
Django

- Competitors: Ruby on Rails, Pyramid etc.
- Large Projects:
  - pinterest.com
  - disqus.com

Django

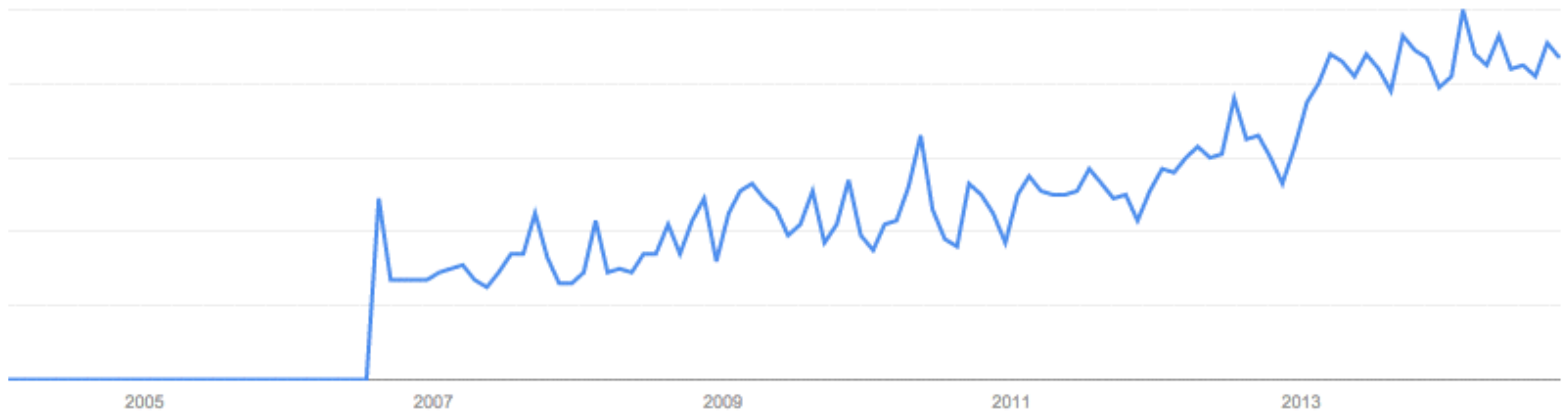
- interest growing! google trends




MHM...MHM
WAIT WHAT?

Django

- That was “Django Unchained” the movie (2013)
- Searching for “python django”

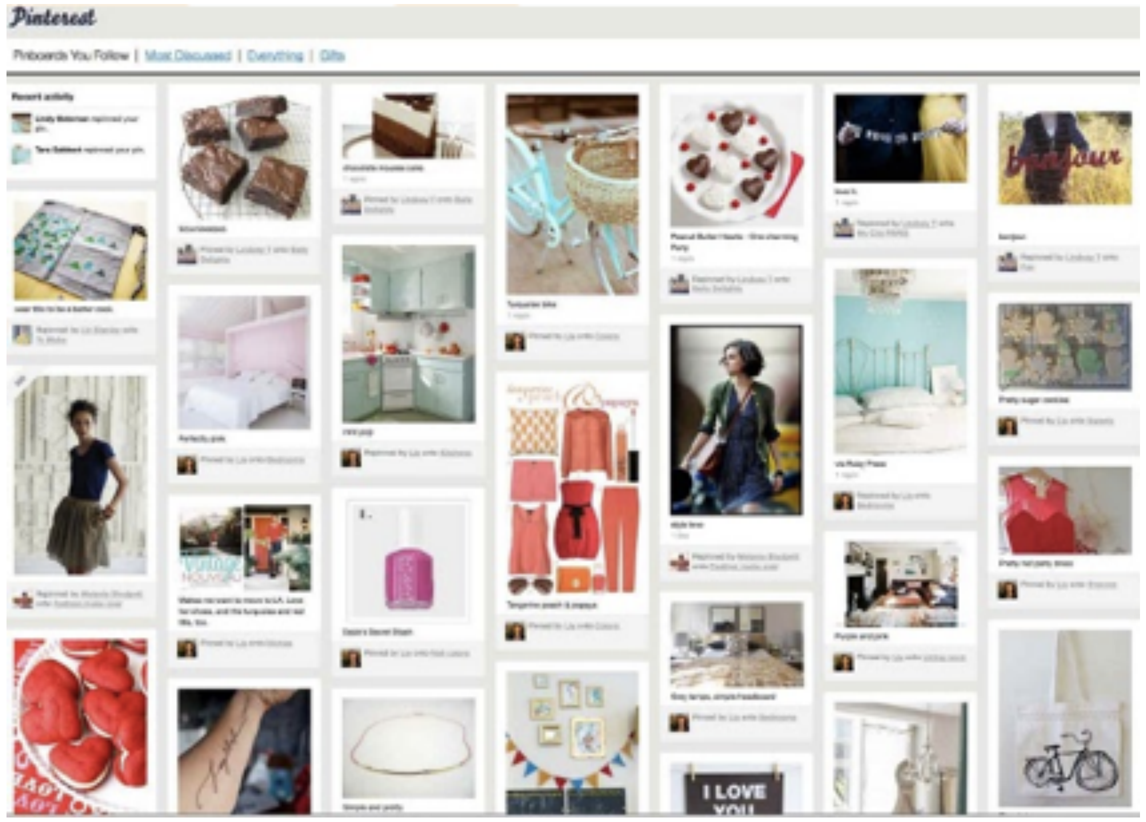


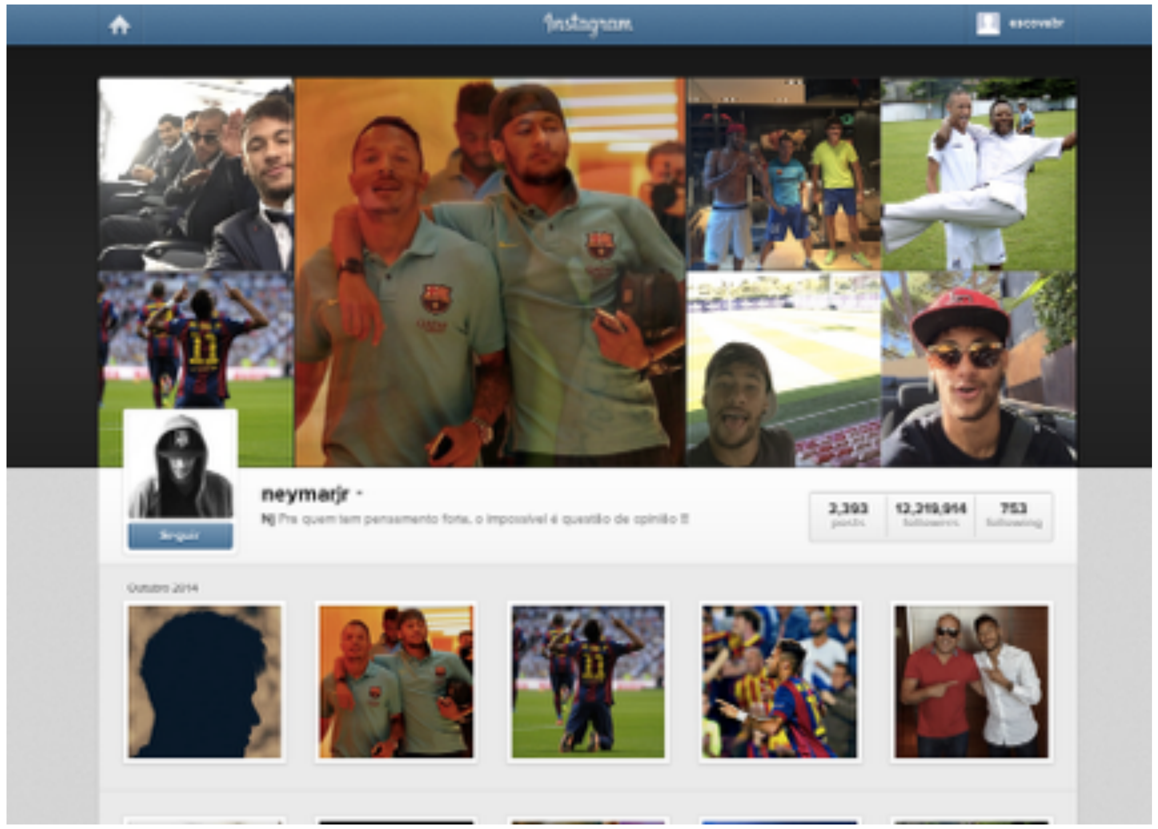
So why django?

- Documentation
- Python
- full-stack framework (lots of batteries built-in)
- Simplifies deploying database backed web applications

Getting Started

Getting Started

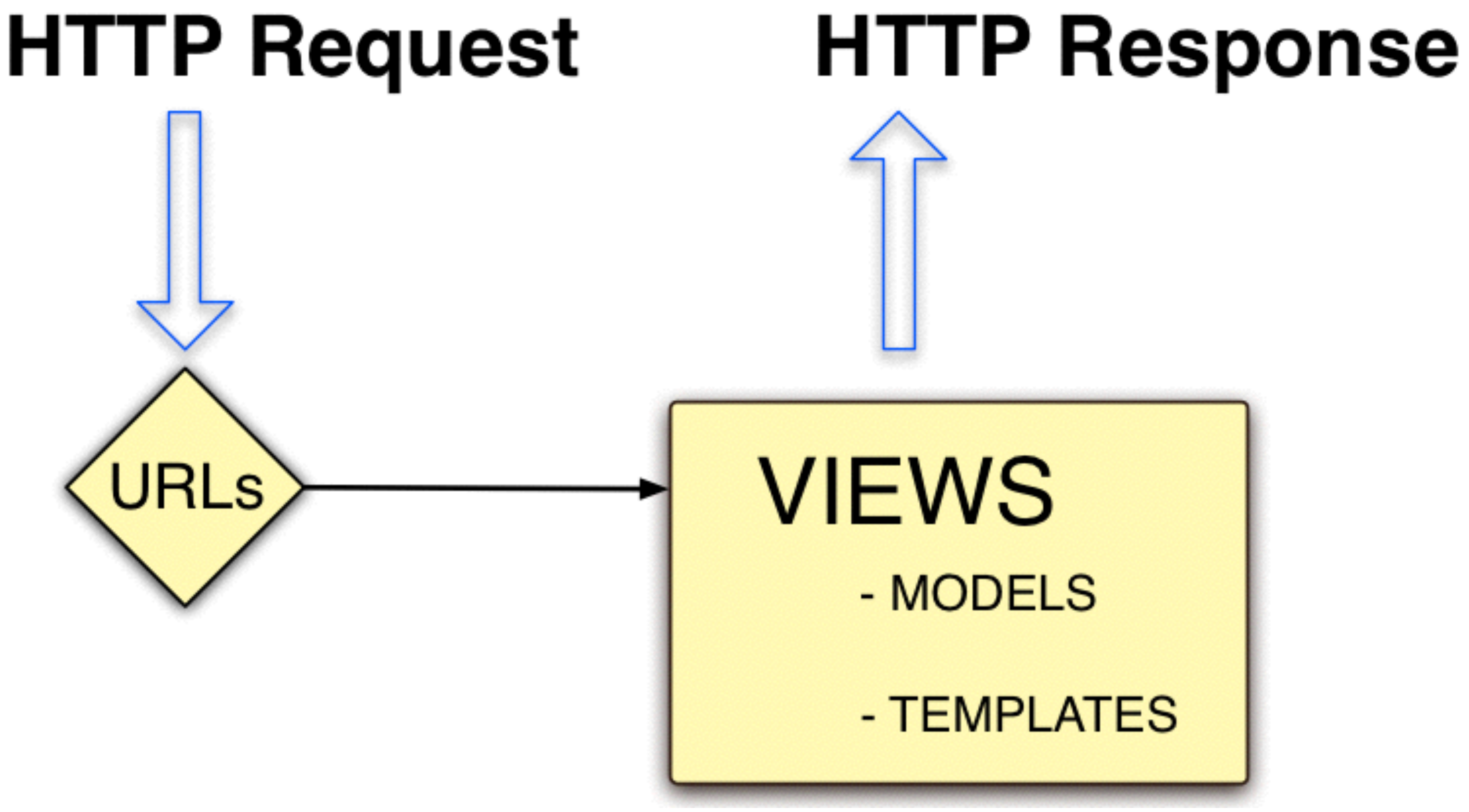




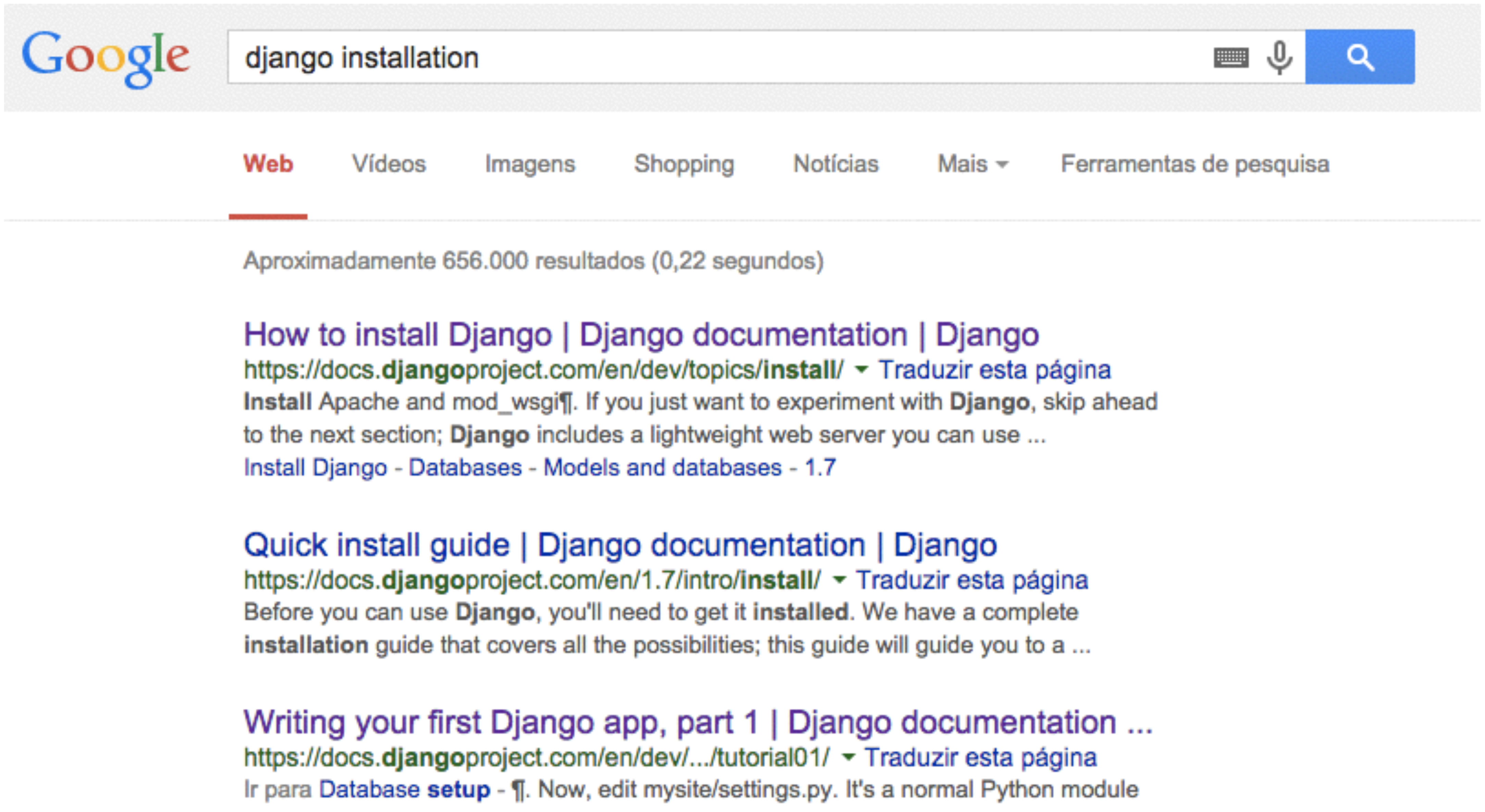
Getting Started

- Instagram and Pinterest are very different web websites;
- But both are at least partly:
  - `HTML interfaces to highly dynamic data stored in a database;`

Django is HTTP in, HTTP out



Django, Installation…



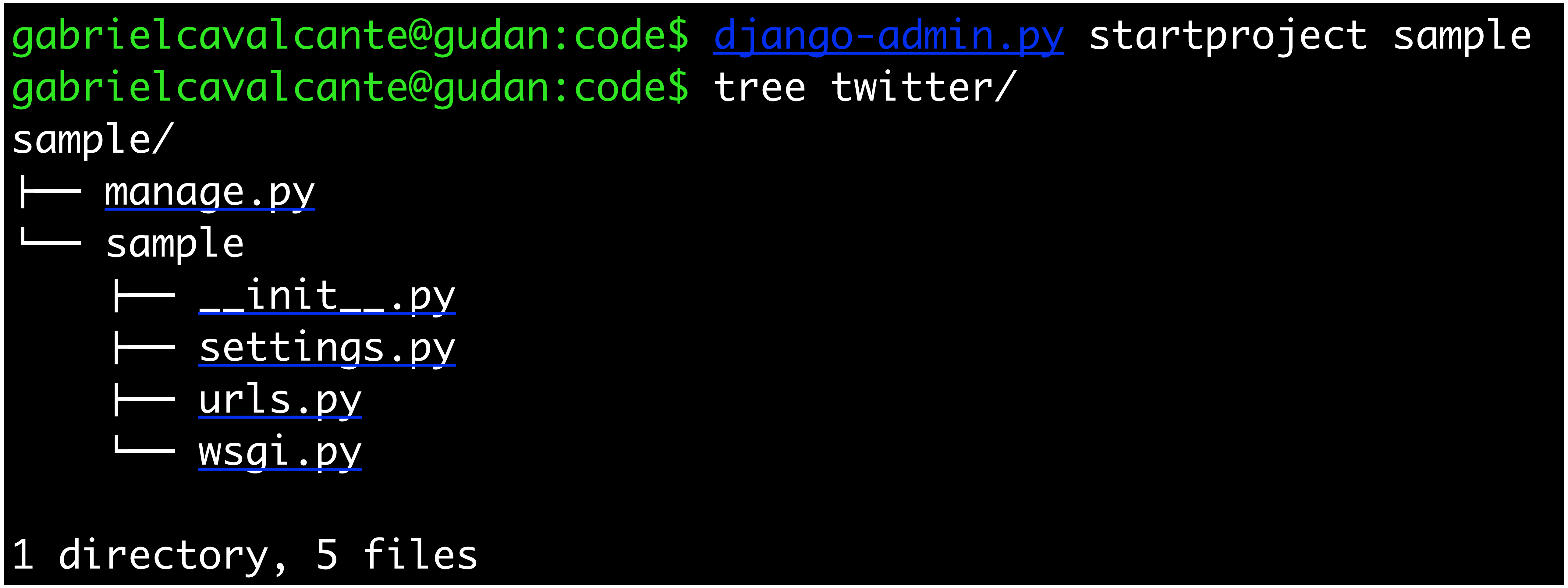
Aproximadamente 656.000 resultados (0,22 segundos)

How to install Django | Django documentation | Django
https://docs.djangoproject.com/en/dev/topics/install/ ▾ Traduzir esta página
Install Apache and mod_wsgi¶. If you just want to experiment with **Django**, skip ahead to the next section; **Django** includes a lightweight web server you can use ...
Install Django - Databases - Models and databases - 1.7

Quick install guide | Django documentation | Django
https://docs.djangoproject.com/en/1.7/intro/install/ ▾ Traduzir esta página
Before you can use **Django**, you'll need to get it **installed**. We have a complete **installation** guide that covers all the possibilities; this guide will guide you to a ...

Writing your first Django app, part 1 | Django documentation ...
https://docs.djangoproject.com/en/dev/.../tutorial01/ ▾ Traduzir esta página
Ir para Database **setup** - ¶. Now, edit mysite/settings.py. It's a normal Python module

Django, Project...

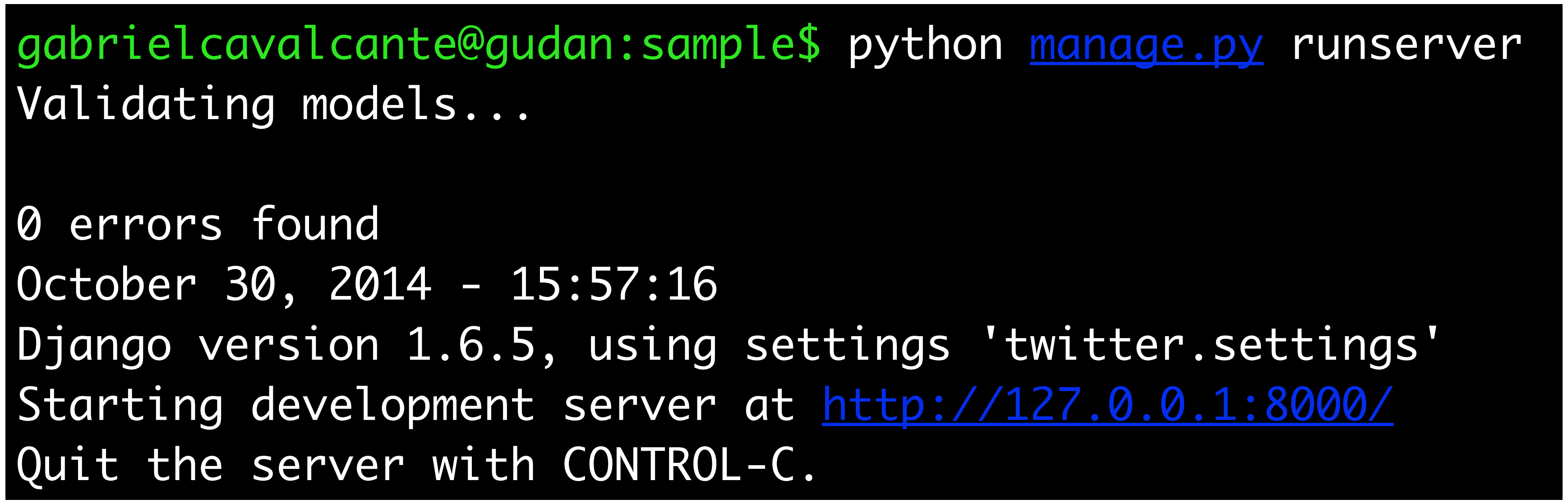


```
gabrielcavalcante@gudan:code$ django-admin.py startproject sample
gabrielcavalcante@gudan:code$ tree twitter/
sample/
├── manage.py
└── sample
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

1 directory, 5 files
```

Django, Project…

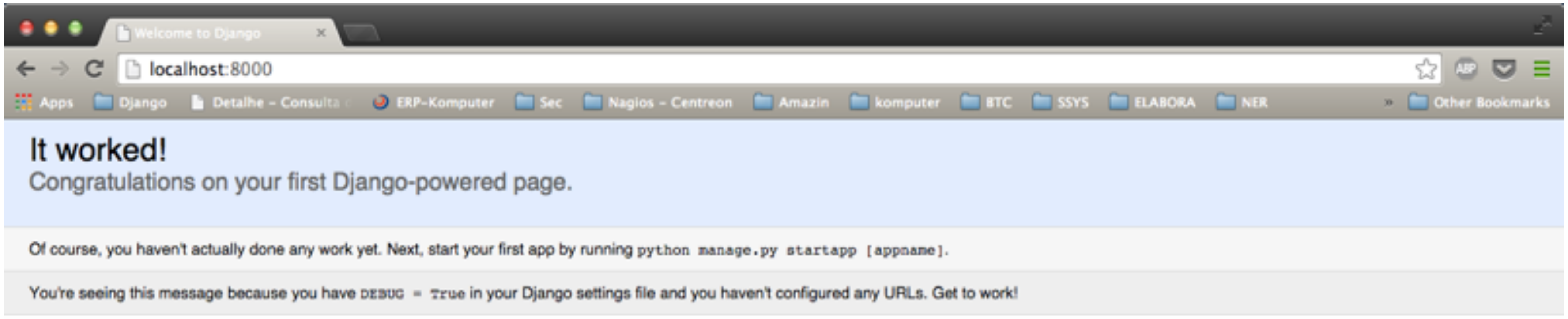
- A project contains:
  - `settings.py` - configuration for the project
  - `manage.py` - command runner
  - `urls.py` - starting point to configure urls

Django, Project, let’s start it



```
gabrielcavalcante@gudan:sample$ python manage.py runserver
Validating models...

0 errors found
October 30, 2014 - 15:57:16
Django version 1.6.5, using settings 'twitter.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Django, Project, let’s start it

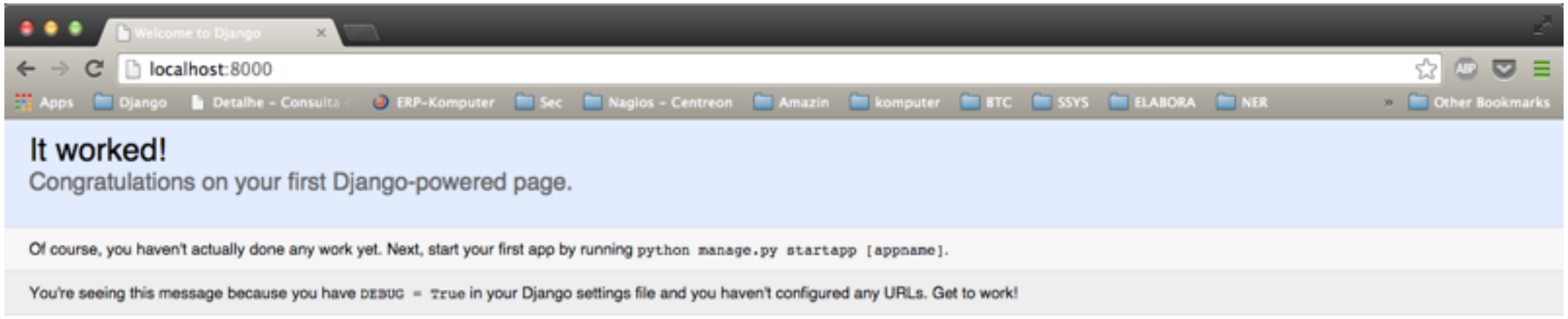


It worked!
Congratulations on your first Django-powered page.

Of course, you haven't actually done any work yet. Next, start your first app by running `python manage.py startapp [appname]`.

You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

Django, Project, let's start it



It worked!
Congratulations on your first Django-powered page.

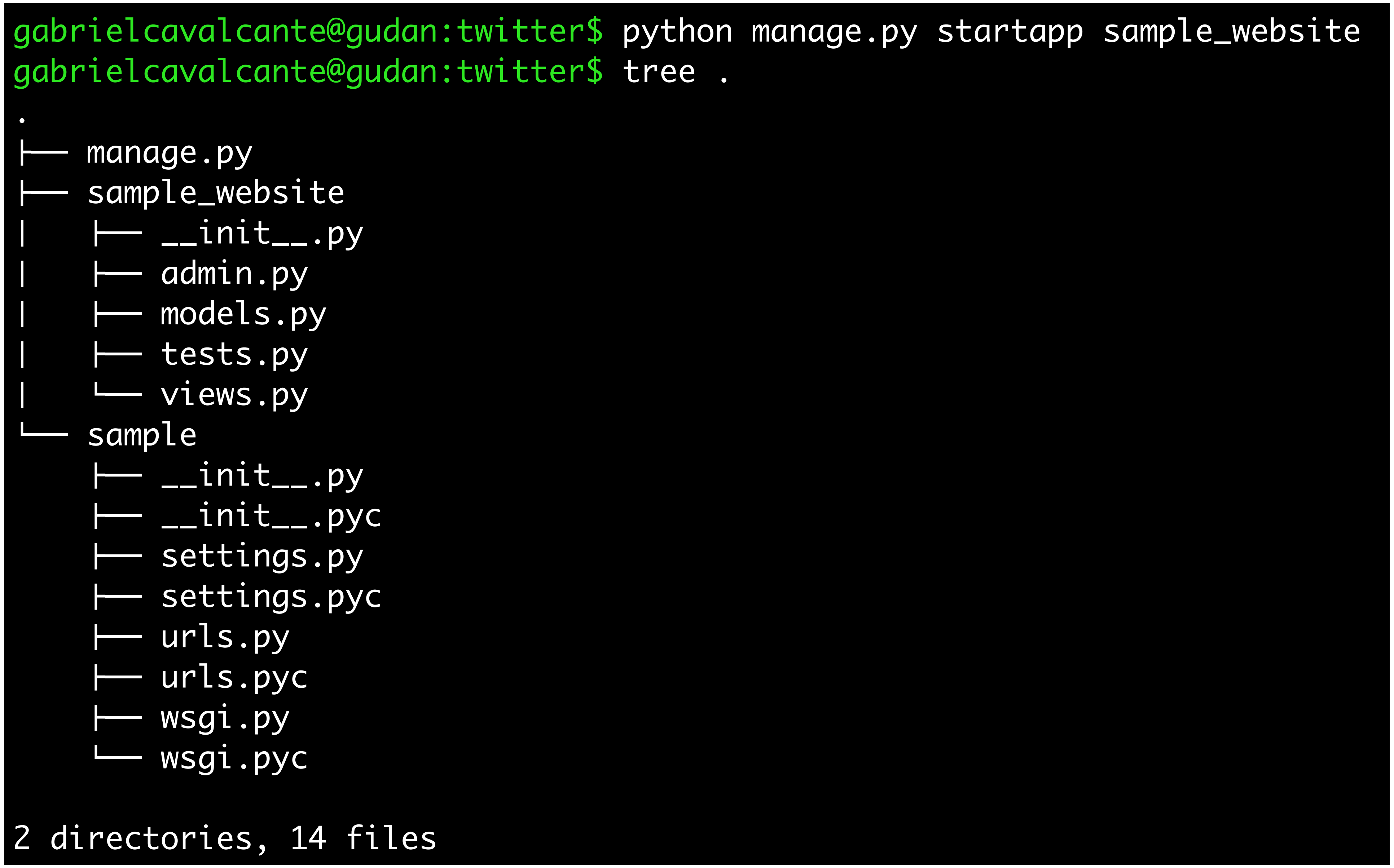
Of course, you haven't actually done any work yet. Next, start your first app by running `python manage.py startapp [appname]`.

You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

Magic!

Django, Project, App

A project is made of many applications
Django includes built in apps in django.contrib (for authentication, security, serving static files)
We need to create our own apps. The start page suggest we try.



```
gabrielcavalcante@gudan:twitter$ python manage.py startapp sample_website
gabrielcavalcante@gudan:twitter$ tree .
.
├── manage.py
├── sample_website
|   ├── __init__.py
|   ├── admin.py
|   ├── models.py
|   ├── tests.py
|   └── views.py
└── sample
    ├── __init__.py
    ├── __init__.pyc
    ├── settings.py
    ├── settings.pyc
    ├── urls.py
    ├── urls.pyc
    ├── wsgi.py
    └── wsgi.pyc

2 directories, 14 files
```

Django, Project, App

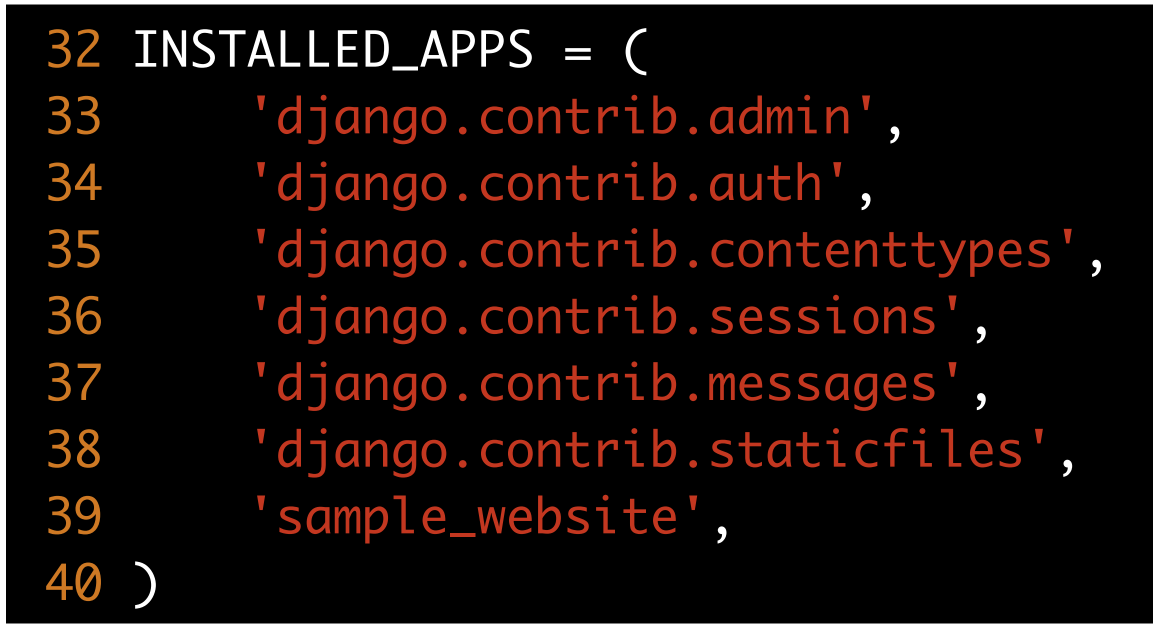
- Take a look at settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Sqlite and others drivers come with Python, so I'll just use that…

Django, Project, App

- Take a look at settings.py (app section)



```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'sample_website',
)
```

The initial project already has many applications, I’ve just added the app that we’ve created a few slides before

Django, Course App

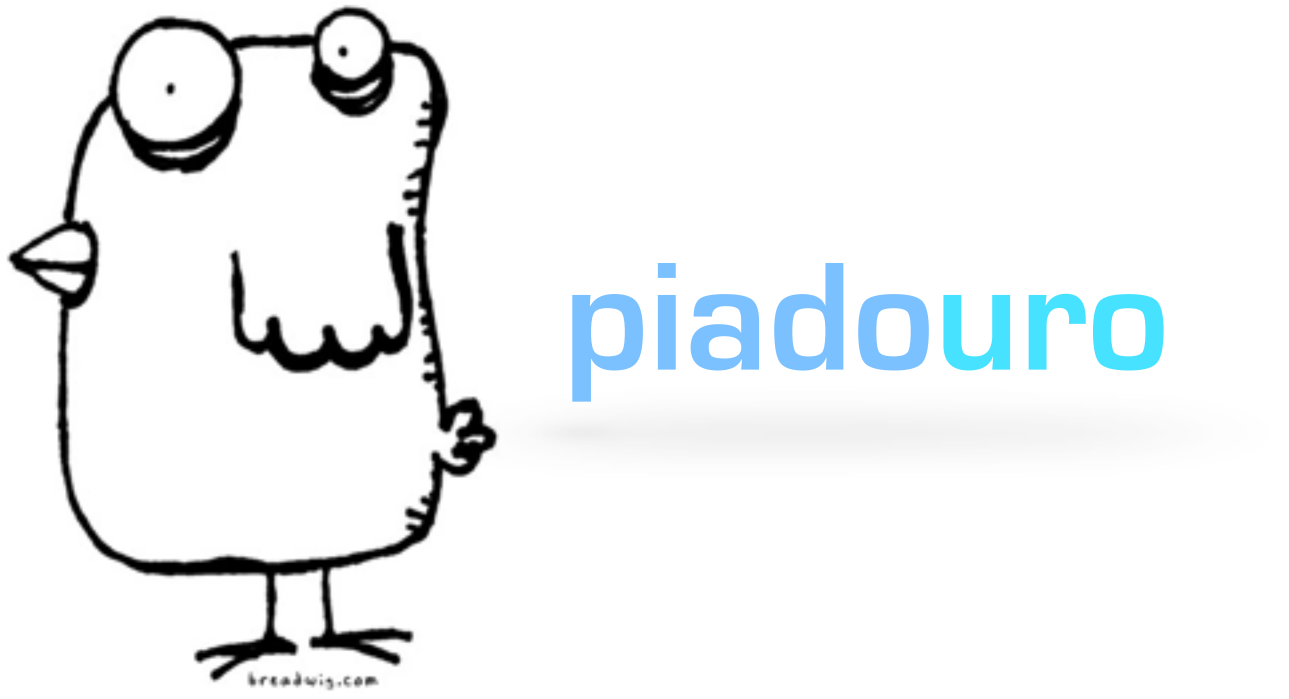
- OK, help me I need some ideas!

- Not bullshit like library system, movie database and others…

?

Django, Course App,

- Start a new project called “piadouro"
- Start a new app called “piadouro_website”
- Run your app and access it in a browser.

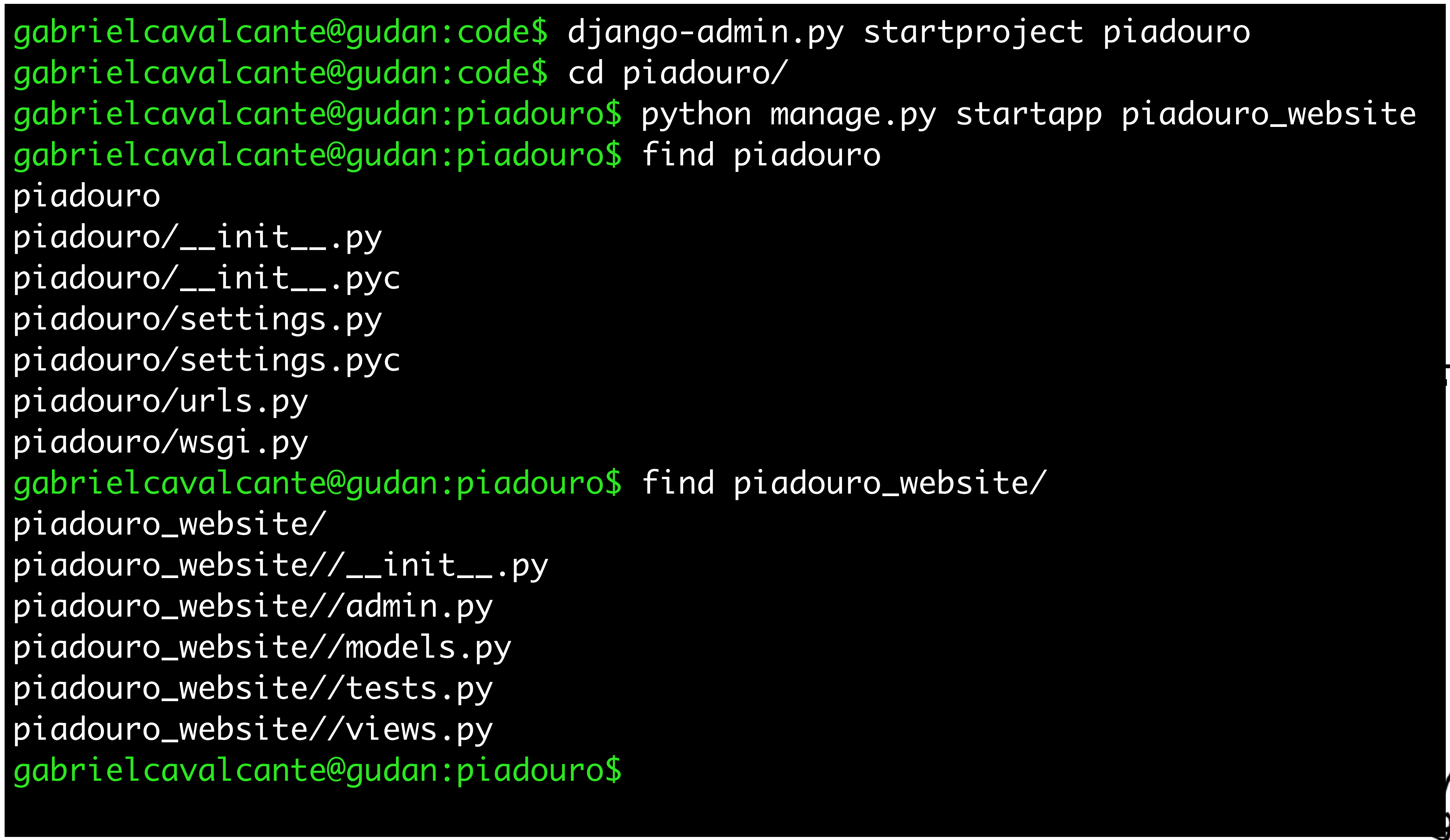


Django, Course App, **Piadouro TO-DO 1:**

- Start a new project called “piadouro"
- Start a new app called “piadouro_website”
- Add “piadouro_website” to INSTALLED_APPS section in settings.py
- Run your app and access it in a browser.
- Take a Look at settings.py

Django, Course App, **Piadouro TO-DO 1:**

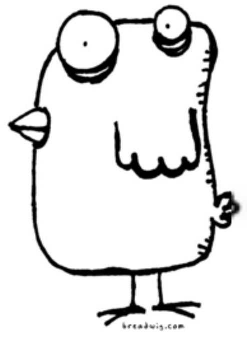
- ~~Start a new project called "piadouro"~~



```
gabrielcavalcante@gudan:code$ django-admin.py startproject piadouro
gabrielcavalcante@gudan:code$ cd piadouro/
gabrielcavalcante@gudan:piadouro$ python manage.py startapp piadouro_website
gabrielcavalcante@gudan:piadouro$ find piadouro
piadouro
piadouro/__init__.py
piadouro/__init__.pyc
piadouro/settings.py
piadouro/settings.pyc
piadouro/urls.py
piadouro/wsgi.py
gabrielcavalcante@gudan:piadouro$ find piadouro_website/
piadouro_website/
piadouro_website//__init__.py
piadouro_website//admin.py
piadouro_website//models.py
piadouro_website//tests.py
piadouro_website//views.py
gabrielcavalcante@gudan:piadouro$
```

Django, #1 urls

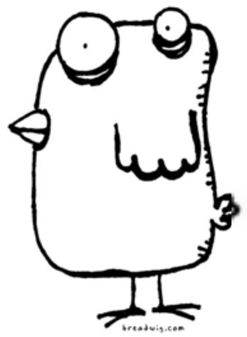
- The part after domain.
- http://fb.com/gdcavalcante/
- All web frameworks: provide a way to map gdcavalcante/ to some code to do something


piadouro

Django, #1 urls

- Django has us write regular expressions that map a url to a view.
- **`urls.py`**

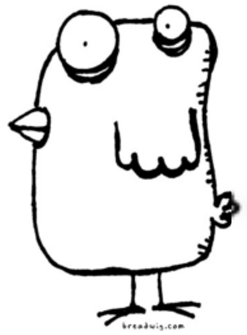
```python
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'piadouro.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^admin/', include(admin.site.urls)),
)
```



Django, #1 urls

- Regular expressions?

- Fast and flexible
- Coming from webserver world of Apache (and others) this the obvious way…
- We can start copying and pasting, so we have time to learn later =)



Django, **`urls.py`**

- Let’s write one url rule

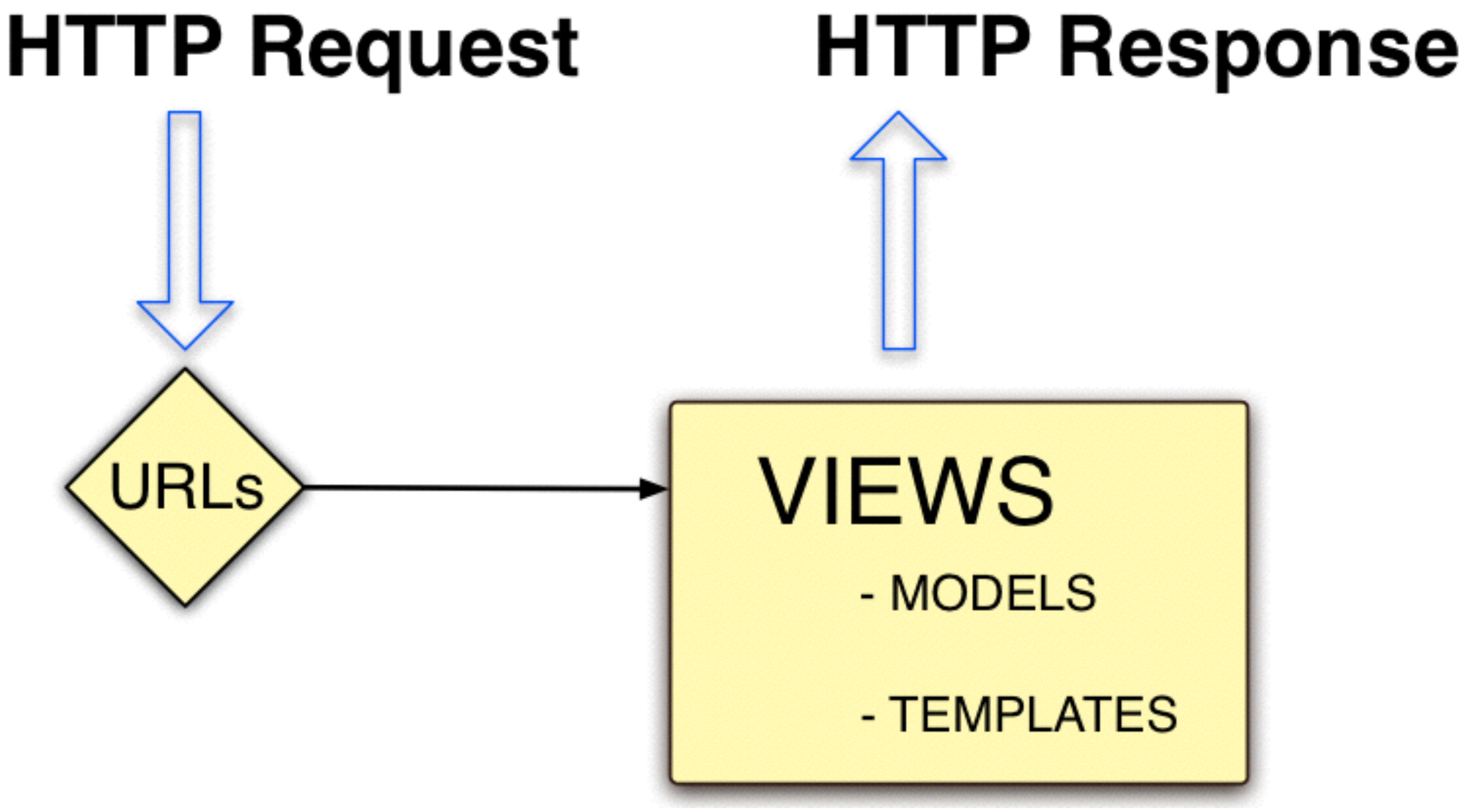
```python
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^$', ‘piadouro_website.views.home’, name='home'),
    #Admin
    url(r'^admin/', include(admin.site.urls)),
)
```

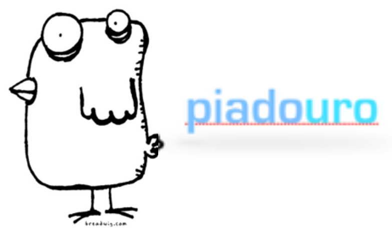
- **`url`** is a function call that builds url patterns
- **`^$`** is the regex for “I didn’t get nothing”
- **`piadouro_website.views.home`** is the Python importing string to get a view
- the name of a url lets us figure out which urls go where later on
- more details: https://docs.djangoproject.com/en/1.7/topics/http/urls/

Django, #2 views

- A view is Python code that takes a `request` object and returns a `response` object.
- Sound familiar?

Django, #2 views





Django is HTTP in, HTTP out

piadouro_website/views.py

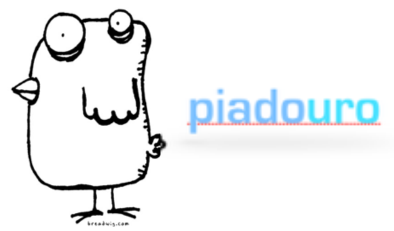
```python
from django.shortcuts import render

# Create your views here.
```

Django is HTTP in, HTTP out

- Defining our home view

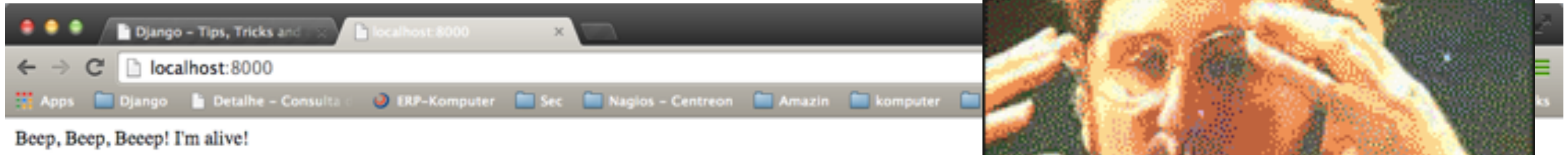
piadouro_website/views.py

```python
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def home(request):
  return HttpResponse("Beep, Beep, Beeep! I'm alive!")
```



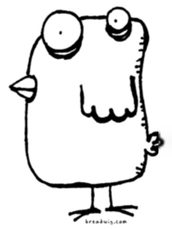
Django is HTTP in, HTTP out

- With manage.py running…



Beep, Beep, Beeep! I'm alive!

- But… I don't see any HTML!



Piadouro TO-DO 2:

- Add a url for homepage at `piadouro/urls.py`

```
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^$', 'piadouro_website.views.home', name='home'),
    #Admin
    url(r'^admin/', include(admin.site.urls)),
)
```

- Add and view to homepage at `piadouro_website/views.py`

```
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def home(request):
  return HttpResponse("Beep, Beep, Beeep! I'm alive!")
```

- Access your web app

Django, #3 templates

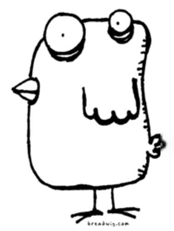
- If we want to make a web application, we better build a response that has some HTML
- My application likely has a lot of html that doesn’t change (the site design) and some that does
- Most web frameworks include template engines
- Basically HTML + placeholders wich might be code.
- Way to break out big chunks of HTML into Separate files.

Django, #3 templates

- Base Template
  - Let’s make a base template with our "site design" in it.

```
gabrielcavalcante@gudan:piadouro$ mkdir -p piadouro/templates/piadouro_website
gabrielcavalcante@gudan:piadouro$ touch piadouro/templates/piadouro_website/base.html
```

- Now we can create a minimal html base template

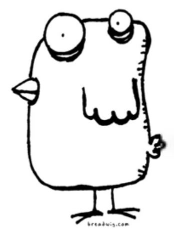


Django, #3 templates

- Define Template location for project

piadouro/settings.py

```
TEMPLATE_DIRS = (
    os.path.join(BASE_DIR, "piadouro/templates"),
)
```



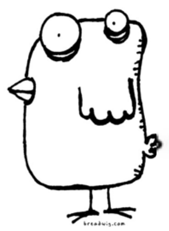
piadouro

Django, #3 templates

piadouro/templates/piadouro_website/base.html

```
<html>
  <head>
  <title>piadouro - the best app</title>
  </head>
  <body>
    <h1>Welcome to piadouro!</h1>
    {% block content %}
    {% endblock %}
  </body>
</html>
```

- All html except for that `{% block %}` part
- Django Template language provides `tags`, `filters`, output and more…


piadouro

Django, #3 templates

- **Template inheritance**
  - Let’s create another template called `home.html`

piadouro/templates/piadouro_website/home.html

```
{% extends “piadouro_website/base.html" %}
{% block content %}
{{ hello }}
{% endblock %}

```

- The extends `block` tag means `home.html` will be shown inside of `base.html`
- So, any blocks in home will show up in the same name block in base
- `{{ hello }}` will output the variable `hello` passed by the view

piadouro

Django, #3 templates

- **Whoa, the view!**

piadouro_website/views.py

```python
from django.shortcuts import render
from django.http import HttpResponse
from django.shortcuts import render_to_response

# Create your views here.
def home(request):
  return render_to_response("piadouro_website/home.html",
                {"hello": "Beep, Beep, Beeep! I'm alive!"})
```

piadouro

Django, #3 templates

- **Whoa, the view!**

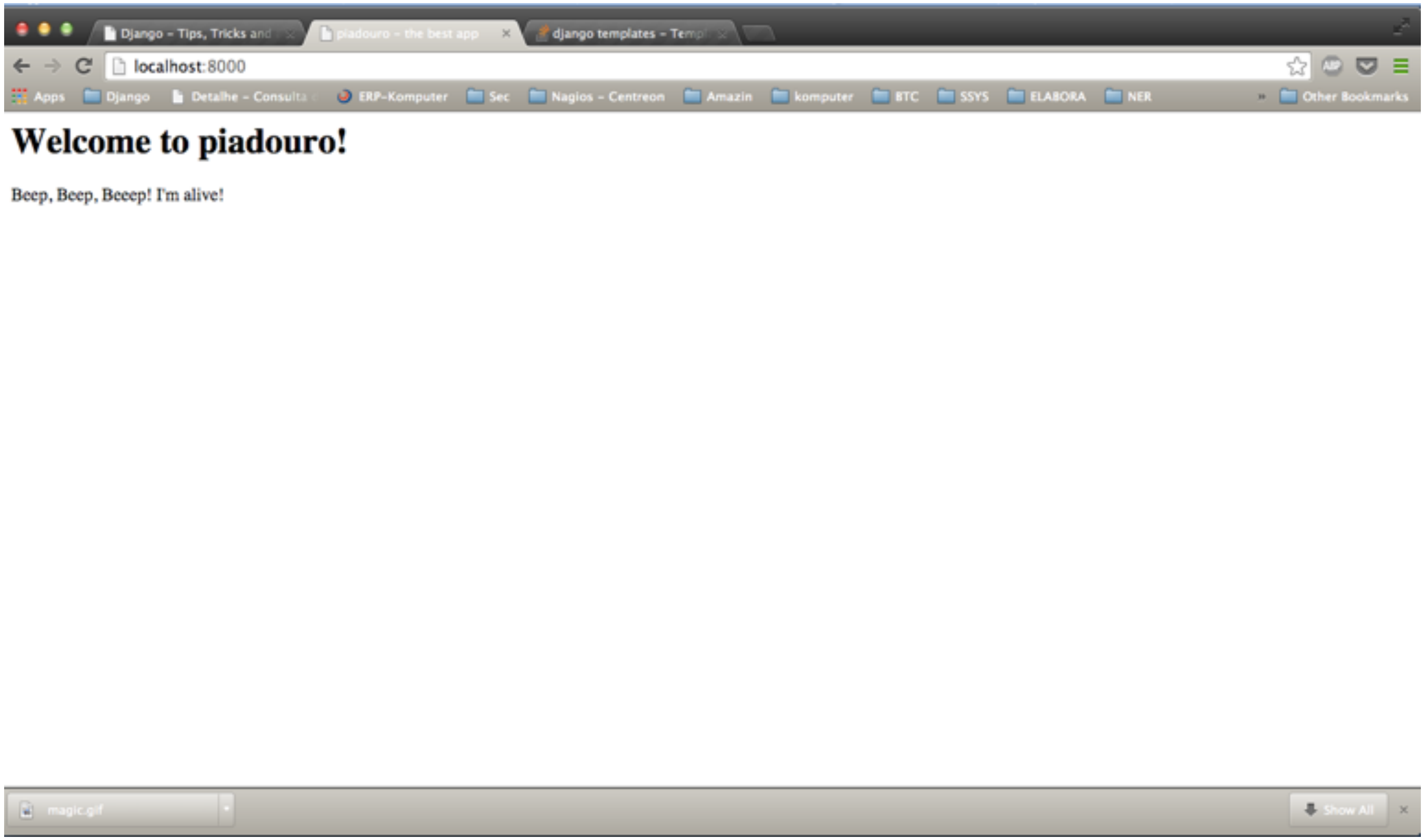
piadouro_website/views.py

```python
from django.shortcuts import render
from django.http import HttpResponse
from django.shortcuts import render_to_response

# Create your views here.
def home(request):
  return render_to_response("piadouro_website/home.html",
                {"hello": "Beep, Beep, Beeep! I'm alive!"})

```

- **Understand?**

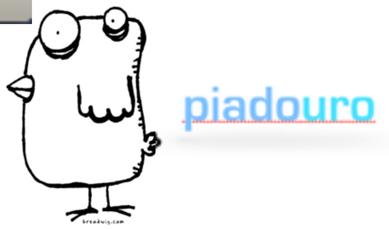
Django, #3 templates

- **App running!**



Welcome to piadouro!

Beep, Beep, Beeep! I'm alive!



Django, Course App, **Piadouro TO-DO 3:**

- Add `home.html` and `base.html`, adjust TEMPLATE_DIRS in `settings.py` and improve home view in `views.py`

piadouro/settings.py

```
TEMPLATE_DIRS = (
    os.path.join(BASE_DIR, "piadouro/templates"),
)
```

piadouro/templates/piadouro_website/base.html

```
<html>
  <head>
  <title>piadouro - the best app</title>
  </head>
  <body>
    <h1>Welcome to piadouro!</h1>
    {% block content %}
    {% endblock %}
  </body>
</html>
```

piadouro/templates/piadouro_website/home.html

```
{% extends “piadouro_website/base.html" %}
{% block content %}
{{ hello }}
{% endblock %}

```

piadouro_website/views.py

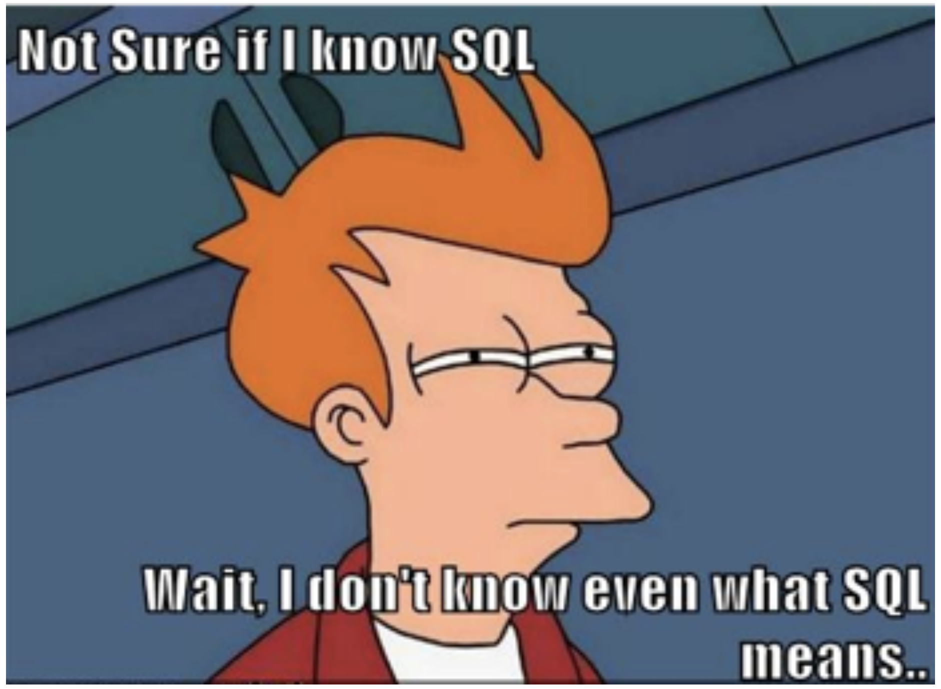
```
from django.shortcuts import render
from django.http import HttpResponse
from django.shortcuts import render_to_response

# Create your views here.
def home(request):
  return render_to_response("piadouro_website/home.html",
                  {"hello": "Beep, Beep, Beeep! I'm alive!"})

```

Django, #3 templates, tags and filters

- You’re going to have to learn a bunch of tags and filters.
- Don’t worry - they’re easy.
- See https://docs.djangoproject.com/en/1.7/topics/templates/ and https://docs.djangoproject.com/en/1.7/ref/templates/builtins/

Django, #4 models

- This is a database backed web app, right?
- We need to Create/Read/Update/Delete data from a relational database.

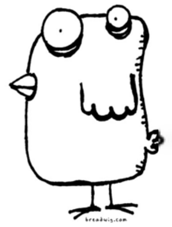


Django, #4 models

- Models: database tables represented in Python Code.
- Your handle to:
  - creating the necessary tables
  - generating the SQL to Create/Read/Update/Delete Data
  - ORM - Object Relational Mapper

Django, #4 models

- Importantly:
  - A model class == Database table.
  - A model instance (object) == a database table row



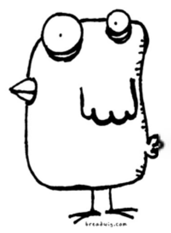
Django, #4 models

- Our first model

piadouro_website/models.py

```
from django.db import models

# Create your models here.

class Piado(models.Model):
  text = models.CharField(max_length=140)

```

- the attributes of the class are columns in the table
- Django automatically takes care of things like PKs if we don't explicitly manage them



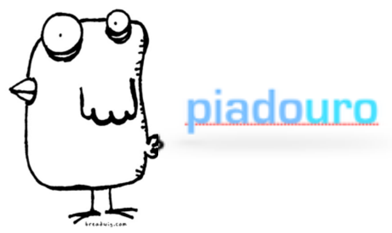
Django, #4 models

- Our first model

piadouro_website/models.py

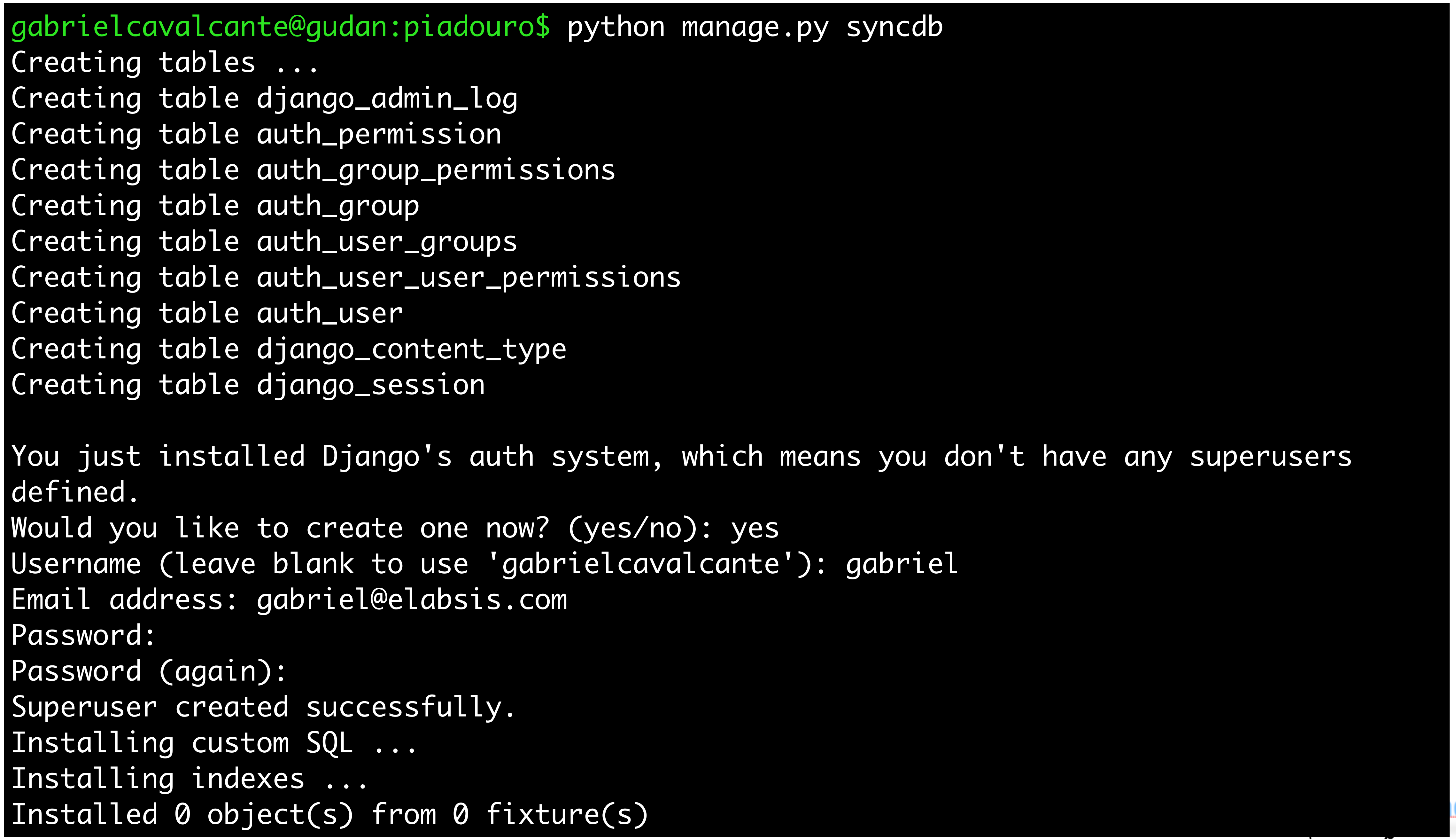
```
from django.db import models

# Create your models here.

class Piado(models.Model):
  text = models.CharField(max_length=140)

```

- the attributes of the class are columns in the table
- Django automatically takes care of things like PKs if we don't explicitly manage them



Django, #4 models

- Django comes with a command to create the database structure for us:



```
gabrielcavalcante@gudan:piadouro$ python manage.py syncdb
Creating tables ...
Creating table django_admin_log
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_groups
Creating table auth_user_user_permissions
Creating table auth_user
Creating table django_content_type
Creating table django_session

You just installed Django's auth system, which means you don't have any superusers
defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'gabrielcavalcante'): gabriel
Email address: gabriel@elabsis.com
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
```

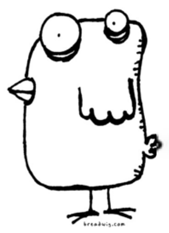
Django, #4 models

- Now that we’ve got a database created for our models let’s try using the interactive Python console to play with them:

```
gabrielcavalcante@gudan:piadouro$ python manage.py shell

In [1]: from piadouro_website.models import Piado

In [2]: p = Piado(text='This is the first Piado ever, really excited #django #python')

In [3]: p.save()
```

- Tha’s all the Python code it takes to create a new row in our database. We can also query our database with python.

```
In [4]: Piado.objects.all()
Out[4]: [<Piado: Piado object>]
```

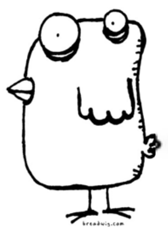


Django, #4 models

- Maybe we should update the view again!

piadouro_website/views.py

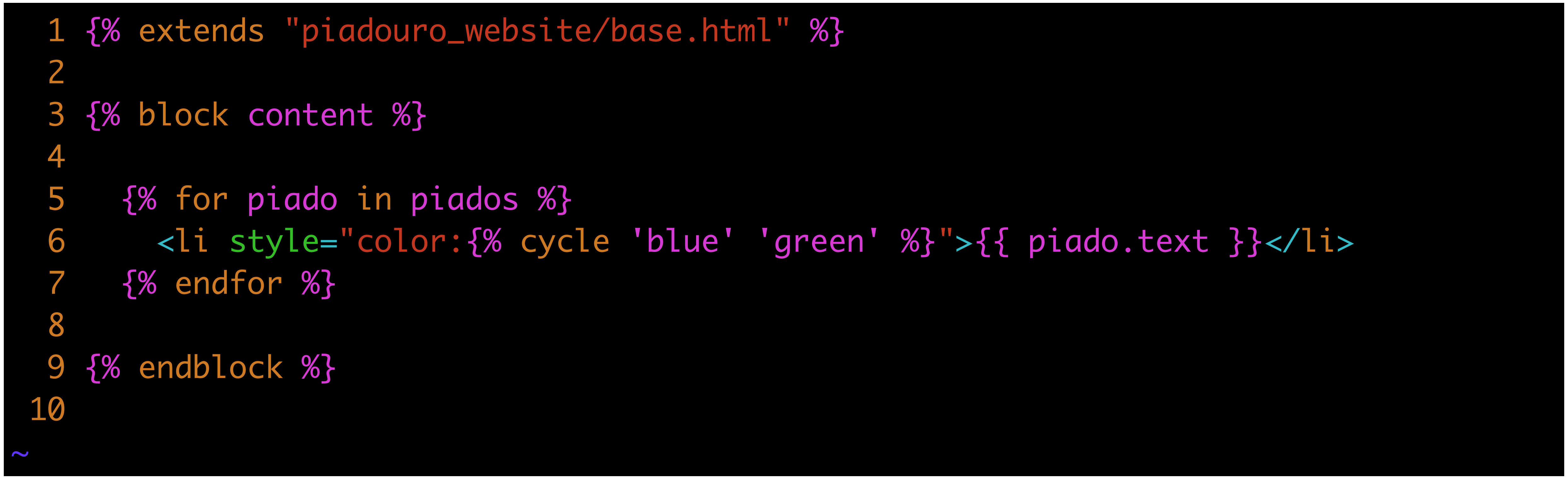
```python
from django.shortcuts import render
from django.http import HttpResponse
from django.shortcuts import render_to_response

#Importing piado_website models
from piadouro_website.models import Piado

# Create your views here.
def home(request):
  return render_to_response("piadouro_website/home.html",
                    { "piados" : Piado.objects.all() })

```



Django, #4 models

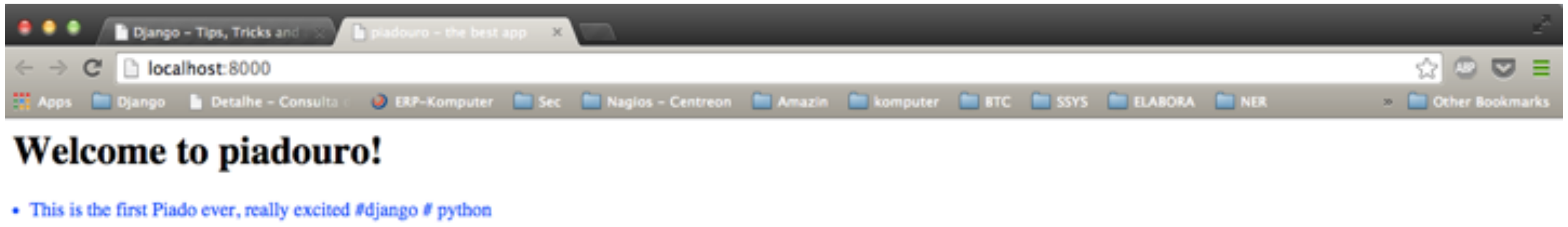
- The template also

piadouro/templates/piadouro_website/home.html



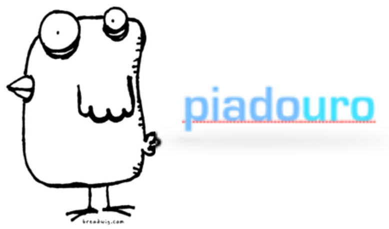
```
{% extends "piadouro_website/base.html" %}

{% block content %}

  {% for piado in piados %}
    <li style="color:{% cycle 'blue' 'green' %}">{{ piado.text }}</li>
  {% endfor %}

{% endblock %}

~
```

Django, #4 models



Welcome to piadouro!

- This is the first Piado ever, really excited #django # python

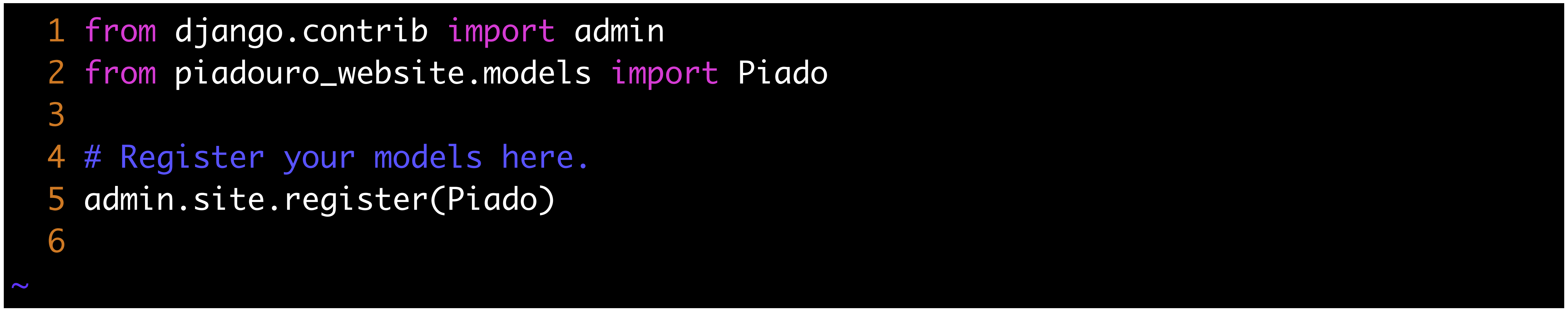


Django, #4 models

- As usual, documentation:
    - https://docs.djangoproject.com/en/1.7/topics/db/models/
    - https://docs.djangoproject.com/en/1.7/topics/db/queries/

Django, #4 models
For free: Admin app

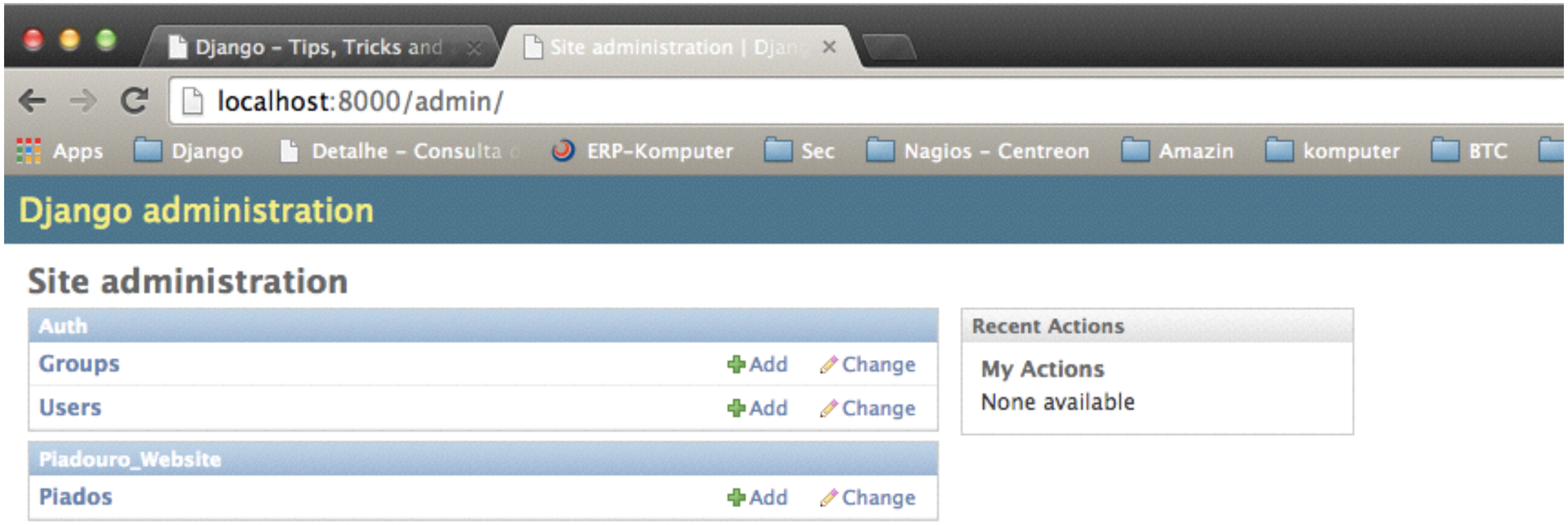
- You can enable the built-in admin app, look at **`urls.py`**, the **`settings.py`**, and by adding an **`admin.py`** file to your application



```
from django.contrib import admin
from piadouro_website.models import Piado

# Register your models here.
admin.site.register(Piado)

~
```

Django, #4 models
For free: Admin app

- You can use the interface to edit data:
- User and password you provided in `syncdb` command


Django - Tips, Tricks and
Site administration | Djan
localhost:8000/admin/
Apps | Django | Detalhe - Consulta | ERP-Komputer | Sec | Nagios - Centreon | Amazin | komputer | BTC

Django administration

Site administration

Auth		
Groups	Add	Change
Users	Add	Change

Piadouro_Website		
Piados	Add	Change

Recent Actions

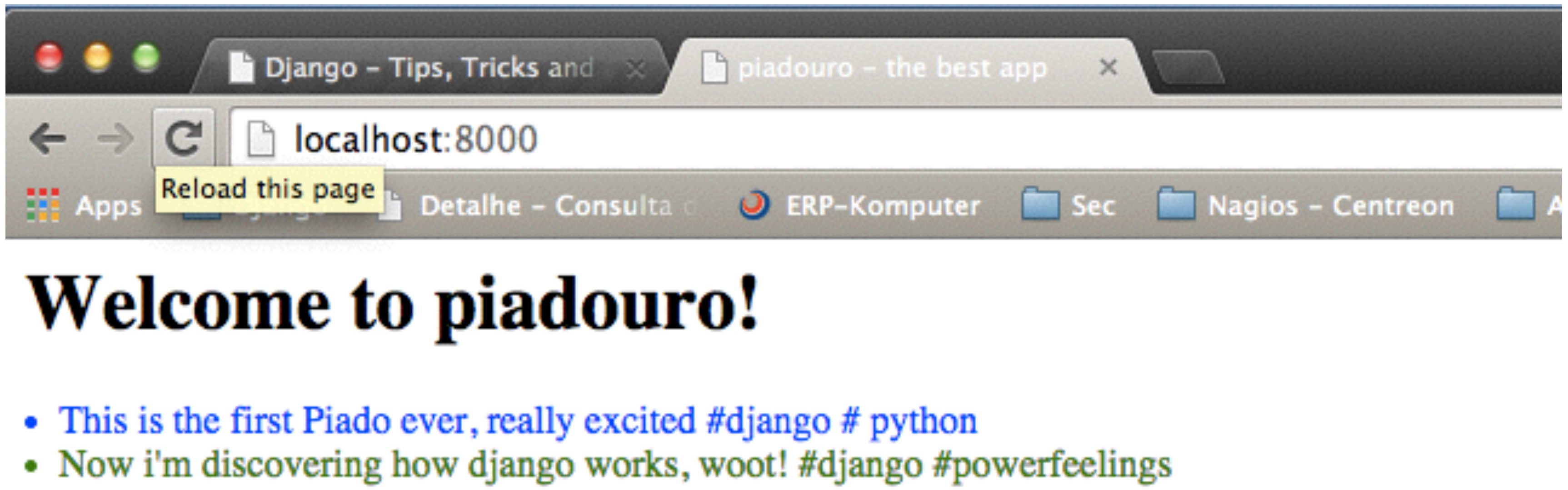
My Actions

None available

piadouro

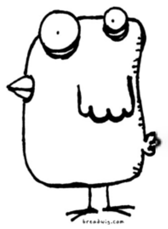
Django, #4 models
For free: Admin app

- Edit data and see results into your website



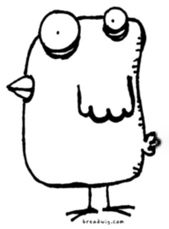
Welcome to piadouro!

- This is the first Piado ever, really excited #django # python
- Now i'm discovering how django works, woot! #django #powerfeelings



Django, #4 models
TO-DO 4:

- Create a Piado class
- Run **`syncdb`** command
- Add some data with python **`manage.py`** shell
- Edit your view to query Piado and pass it to template
- Edit your template to **`iterate`** over Piado objects and create html nodes
- Activate admin interface to Piado class
- Edit some data and see the results into website

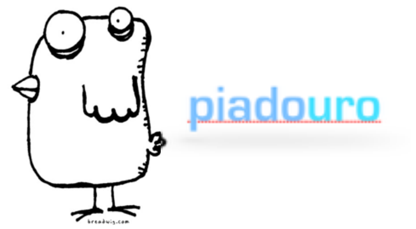


Django, #5 forms

- Usually website uses forms to achieve data from users;
- Django has to form generating classes: forms.Form and forms.ModelForm
- The field rendering into html form can be done by the template tags: `form.as_p`, `form.as_li` or `form.as_table`;

Django, #5 forms

- Fields:
  - map data between forms (received as a POST request) and django models;
- Widgets:
  - The visual aspects of each field in form html (e.g. DateField or TextInput for dates);
- https://docs.djangoproject.com/en/1.7/ref/forms/widgets/

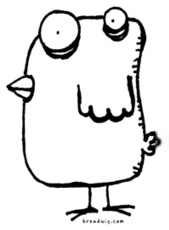


Django, #5 forms

- Use Meta class to easy build forms:

piadouro_website/forms.py

```python
from django import forms
from piadouro_website.models import Piado


class FormItemPiado(forms.Form):
  class Meta:
    model = Piado
```



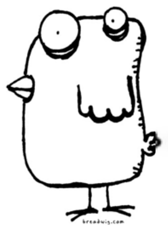
Django, #5 forms

- Create a view to render the form

piadouro_website/views.py

```python
from piadouro_website.forms import FormItemPiado
```

```python
def piado_add(request):
  if request.method == 'POST':
    form = FormItemPiado(request.POST,request.FILES)
    if form.is_valid():
      form.save()
      return HttpResponseRedirect('/')
    else:
      pass
  else:
    form = FormItemPiado()
  return render(request,'piadouro_website/new_piado.html', {'form' : form})

```



Django, #5 forms

- Create an url entry to the form

piadouro_website/urls.py

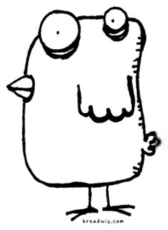
```
url(r'^newpiado/$', 'piadouro_website.views.piado_add', name='new_piado'),
```

Django, #5 forms

- Create a new template to render the form

piadouro/templates/piadouro_website/new_piado.html

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

{% block content %}
<form action="" method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Send</button>
</form>
{% endblock %}
```



piadouro_website/urls.py

```python
url(r'^newpiado/$', 'piadouro_website.views.piado_add', name='new_piado'),
```

piadouro/templates/piadouro_website/new_piado.html

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

{% block content %}
<form action="" method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Send</button>
</form>
{% endblock %}
```

piadouro_website/forms.py

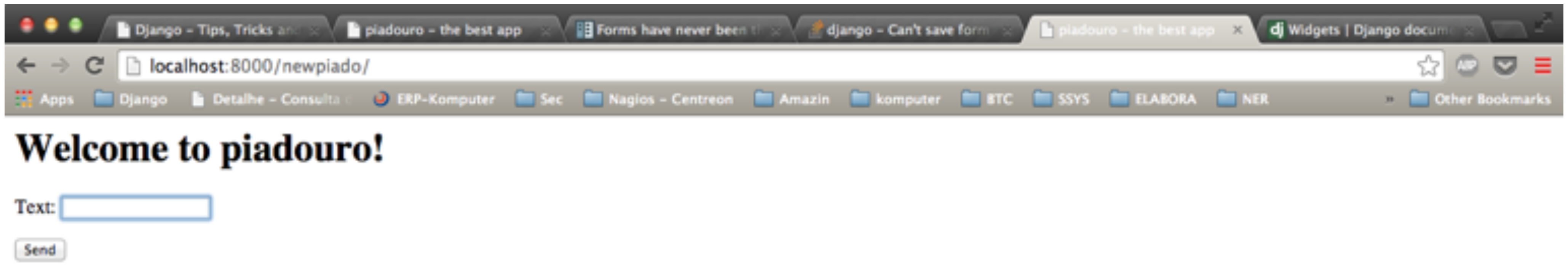
```python
from django import forms
from piadouro_website.models import Piado


class FormItemPiado(forms.ModelForm):
  class Meta:
    model = Piado
```

piadouro_website/views.py

```python
from piadouro_website.forms import FormItemPiado
from django.http import HttpResponse,HttpResponseRedirect
def piado_add(request):
  if request.method == 'POST':
    form = FormItemPiado(request.POST,request.FILES)
    if form.is_valid():
      form.save()
      return HttpResponseRedirect('/')
    else:
      pass
  else:
    form = FormItemPiado()
  return render(request,'piadouro_website/new_piado.html', {'form' : form})

```

Django, #5 forms

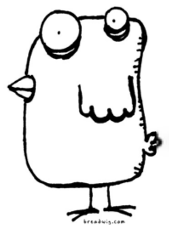
- Running the server
- The Form rendered



Welcome to piadouro!

Text:

Send



Django, #5 forms
TODO 5:

- Create the file forms.py and define a new form to Piado Model.
- Create a new view to render the form and save a new posted form
- Create a new template to render the form
- Create a new url entry to access the view

Django, Let’s Hack

- How to connect User to tweet?

Django, Let’s Hack

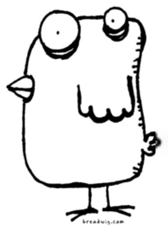
- Application: django.contrib.auth
  - Uses:
    - django.contrib.sessions
    - django.contrib.contenttypes
- *Models*:
  - User
  - Group
  - Permission
- Profile System

Django, Let’s Hack

- Add User Foreign key into our Piado class:

piadouro_website/models.py

```python
from django.db import models
from django.contrib.auth.models import User
# Create your models here.

class Piado(models.Model):
  text = models.CharField(max_length=140)
  user = models.ForeignKey(User)

```



Django, Let's Hack

- TODO:

piadouro_website/views.py

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

```
@login_required
def piado_add(request):
  if request.method == 'POST':
    form = FormItemPiado(request.POST,request.FILES)
    if form.is_valid():
      piado = form.save(commit=False)
      piado.user = request.user
      piado.save()
      return HttpResponseRedirect('/')
    else:
      pass
  else:
    form = FormItemPiado()
  return render(request,'piadouro_website/new_piado.html', {'form' : form})
```

Django, Let's Hack

- Add a new url entry to login page

piadouro/urls.py

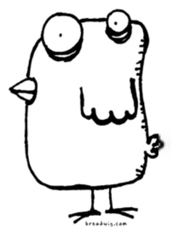
```python
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^$', 'piadouro_website.views.home', name='home'),
    url(r'^newpiado/$', 'piadouro_website.views.piado_add', name='new_piado'),
    url(r'^accounts/login/$', 'django.contrib.auth.views.login', {'template_name': 'admin/login.html'},name="my_login"),
    #Admin
    url(r'^admin/', include(admin.site.urls)),
)
```

Django, Let's Hack

- Adjust form to ask only text field

piadouro/forms.py

```python
from django import forms
from piadouro_website.models import Piado


class FormItemPiado(forms.ModelForm):
  class Meta:
    model = Piado
    fields = ['text']

```



Django, Let's Hack

- Add user FK to our Piado model
- Adjust newpiado view to save the user coming from request
- Add an url to use django.admin login page
- Uses Admin page to create new user into django app
- Just figure out by yourself:
  - Requires login to home page;
  - How to include an logout link? (tip: search for django admin logout page);
  - Create a another view like home, to see only your own piados; * maybe called **/mypiados =)**

Django, Let’s Hack #2

- In twitter people can follow,
- Add a new model called “Follow”. It will be responsible for track users that you follow.
- You have to use a new kind of field called “ForeignKey"
- Django will map that relation for us, and will apply in database schema.

Django, Let’s Hack #2

- Adjust your models

piadouro/models.py

```python
from django.db import models
from django.contrib.auth.models import User
# Create your models here.

class Piado(models.Model):
  text = models.CharField(max_length=140)
  user = models.ForeignKey(User)

class Follow(models.Model):
  follower_user = models.ForeignKey(User,related_name="follower")
  followed_user = models.ForeignKey(User,related_name="followed")
```

- Don’t forget to remove your actual database (db.sqlite3) and re-run **`python manage syncdb`**

Django, Let's Hack #2

- Don't forget a view to list users:

piadouro/views.py

```
@login_required
def users(request):
  return render_to_response("piadouro_website/users.html",
                      { "users" : User.objects.all(), "user":
request.user} )
```

- And the template:

piadouro/templates/piadouro_website/users.html

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

{% block content %}

  {% for u in users %}
    <div align="center">
      <p style="color:{% cycle 'blue' 'green' %}">{{ u }}</p>
    </div>
  {% endfor %}

{% endblock %}
```

Django, Let’s Hack #2

- We also need an url entry:

piadouro/urls.py

```
url(r'^users/$', 'piadouro_website.views.users', name='users'),
```

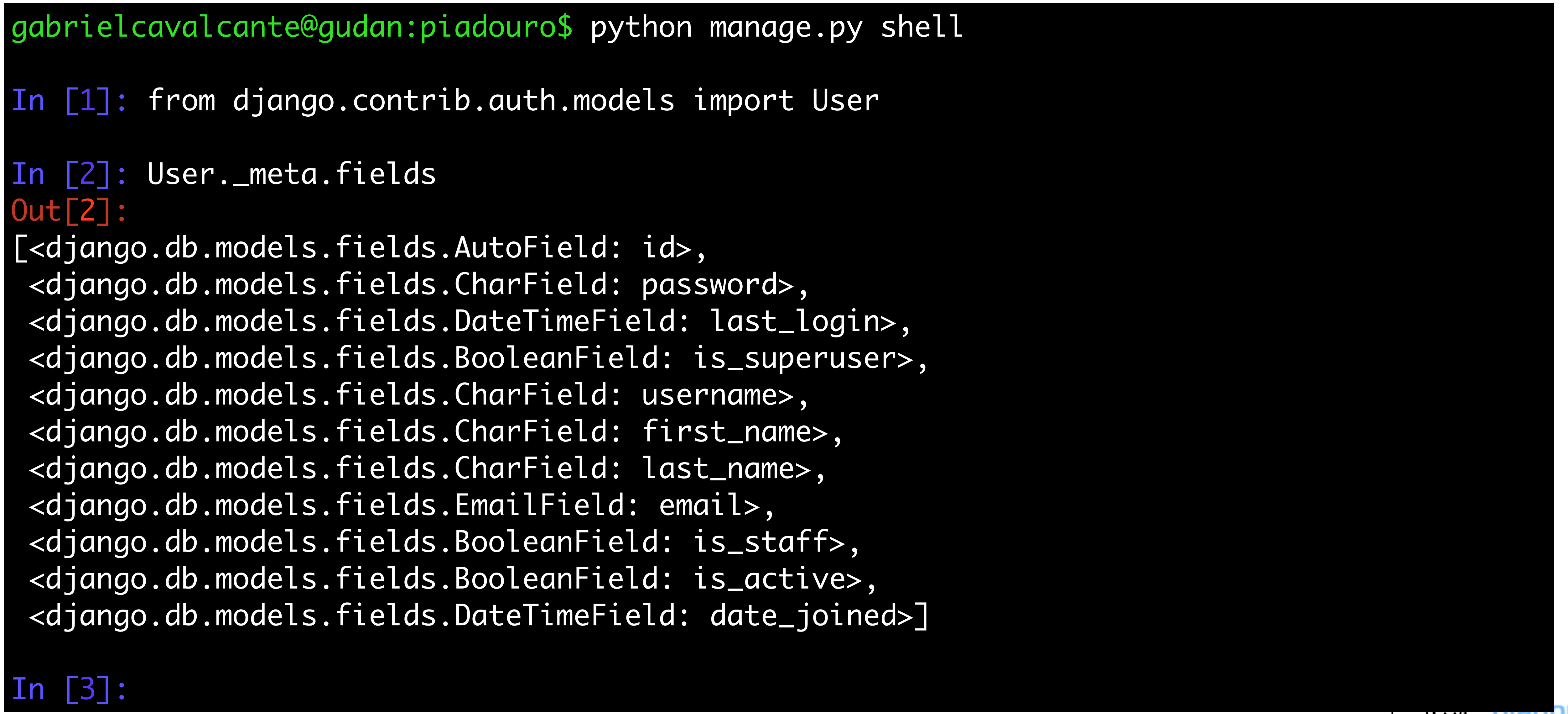
- How about to adjust our base template to show username and links for other pages

piadouro/templates/piadouro_website/base.html

```
<html>
  <head>
  <title>piadouro - the best app</title>
  </head>
  <body>
    <div align="right">
     Hello, <b>{{ user }}</b> <a href="/">Home</a> <a href="/mypiados">My Piados</a>
<a href="    /logout">logout</a>
    </div>
    <h1>Welcome to piadouro!</h1>
    {% block content %}
    {% endblock %}
  </body>
</html>

```

Django, URL parameters

- So now, we have multiple users. And also the users need a profile page. Let's take a look at User class



```
gabrielcavalcante@gudan:piadouro$ python manage.py shell

In [1]: from django.contrib.auth.models import User

In [2]: User._meta.fields
Out[2]:
[<django.db.models.fields.AutoField: id>,
 <django.db.models.fields.CharField: password>,
 <django.db.models.fields.DateTimeField: last_login>,
 <django.db.models.fields.BooleanField: is_superuser>,
 <django.db.models.fields.CharField: username>,
 <django.db.models.fields.CharField: first_name>,
 <django.db.models.fields.CharField: last_name>,
 <django.db.models.fields.EmailField: email>,
 <django.db.models.fields.BooleanField: is_staff>,
 <django.db.models.fields.BooleanField: is_active>,
 <django.db.models.fields.DateTimeField: date_joined>]

In [3]:
```

Django, URL parameters

- How we can create an url for each user?

```
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    #Piados and home page
    url(r'^$', 'piadouro_website.views.home', name='home'),
    url(r'^mypiados$', 'piadouro_website.views.mypiados', name='mypiados'),
    url(r'^newpiado/$', 'piadouro_website.views.piado_add', name='new_piado'),
    #Users pages
    url(r'^users/$', 'piadouro_website.views.users', name='users'),
    url(r'^users/(?P<username>\w+)/','piadouro_website.views.profile',name='profile'),
    #Login/Logout
    url(r'^accounts/login/$', 'django.contrib.auth.views.login', {'template_name': 'admin/login.html'},name="my_login"),
    url(r'^logout$', 'django.contrib.auth.views.logout', {'template_name': 'piadouro_website/home.html'},name="my_logout"),

    #Admin
    url(r'^admin/', include(admin.site.urls)),
)
```

- The pattern in url will be a parameter received by the view

Django, URL parameters

- Create a view to get user object and pass to a template

piadouro_website/views.py

```
from django.shortcuts import render_to_response, get_object_or_404
```

```
@login_required
def profile(request,username):
  #Get the user object
  user = get_object_or_404(User,username=username)
  #Get all piados
  piados = Piado.objects.filter(user=user)
  return render(request,"piadouro_website/profile.html",
{'u':user,'piados':piados})
```

Django, URL parameters

- Create a new template (tip: use home/mypiados as initial).

piadouro/templates/piadouro_website/profile.html

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

{% block content %}
  <div id="user_info" align="left">
    <a href="/users/{{ u.username }}/">@{{ u.username }}</a>
    <i>{{ u.first_name }} {{ u.last_name }}</i>
    <br>
    <i>{{ u.email }}</i>
    <br>
    Joined: {{ u.date_joined }}
  </div>
  <div style="padding:40px;">
  {% for piado in piados %}
    <li style="color:{% cycle 'blue' 'green' %}">{{ piado.text }}</li>
  {% endfor %}
  </div>
{% endblock %}
```

Django, URL parameters

- Don’t forget to add links and extra info to profiles in users list page

piadouro/templates/piadouro_website/users.html

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

{% block content %}

  {% for u in users %}
   <div id="user_info" style="padding:10px;" align="left">
    <a href="/users/{{ u.username }}/">@{{ u.username }}</a>
    <i>{{ u.first_name }} {{ u.last_name }}</i>
    <br>
    <i>{{ u.email }}</i>
    <br>
    Joined: {{ u.date_joined }}
  </div>
  {% endfor %}
{% endblock %}
```

Django, URL parameters

- Todo:
  - Add a profile page (view, url and template)
  - Adjust users list page to show more info an links to profile page

  - Don’t forget to add “Users list” link into base.html

Django, Adding a follow button

- Let's just add a follow button into the users profile page
- Modify the view to check if a user is followed (use the url parameter passed by url function)
- Use django tags to choose what button we should enable

Django, URL parameters

- Add a variable to pass to rendering engine, so it can decide wether put follow or unfollow form

piadouro_website/views.py

```
from piadouro_website.models import Piado,Follow
```

```
@login_required
def profile(request,username):
  #Get the user object
  user = get_object_or_404(User,username=username)
  #Get all piados
  piados = Piado.objects.filter(user=user)
  #See if the user is a follow
  followed= len(Follow.objects.filter(followed_user=user,
                                      follower_user=request.user)) > 0
  return render(request,"piadouro_website/profile.html",
                         {‘u':user,'piados':piados, 'followed': followed})
```

piadouro

Django, URL parameters

- Adjust our profile template to create custom buttons in some place

piadouro/templates/piadouro_website/profile.html

```
    {% if followed %}
    <form action="follow" method="get">
      <input type="submit" name="Follow" value="Unfollow" />
    </form>
    {% else %}
    <form action="follow" method="get">
      <input type="submit" name="Follow" value="Follow" />
    </form>
    {% endif %}
```

Django, URL parameters

- That’s sufficient????

Django, URL parameters

- NOOOO! We've to handle the request.POST from buttons

piadouro_website/views.py

```python
@login_required
def profile(request,username):
  #Get the user object
  user = get_object_or_404(User,username=username)
  #Get all piados
  piados = Piado.objects.filter(user=user)
  if 'Follow' in request.GET:
    if request.GET['Follow'] == 'Follow':
      follow = Follow()
      follow.followed_user = user
      follow.follower_user = request.user
      follow.save()
    else:
      follow = Follow.objects.filter(followed_user__username=username,
                          follower_user=request.user)[0]
      follow.delete()
    return HttpResponseRedirect('/users/'+user.username+'/')
  #See if the user is a follow
  followed= len(Follow.objects.filter(followed_user__username=username,
                                       follower_user=request.user)) > 0
  return render(request,"piadouro_website/profile.html",{'u':user,'piados':piados,
                                                          'followed': followed})
```

Django, URL parameters

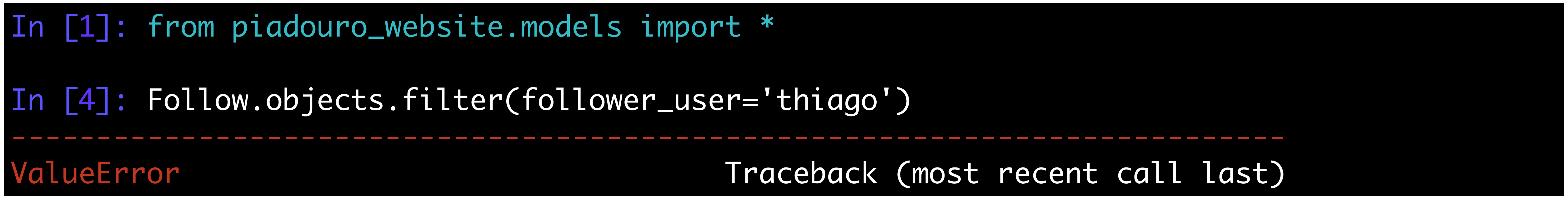
- Todo:
  - Adjust the profile view to send a boolean flag to show follow/unfollow button
  - Adjust the template to use the boolean flag coming from the view
  - Readjust the profile view to create/delete follow objects

Django, Querysets

- How to filter my home to show only Piados from followed users

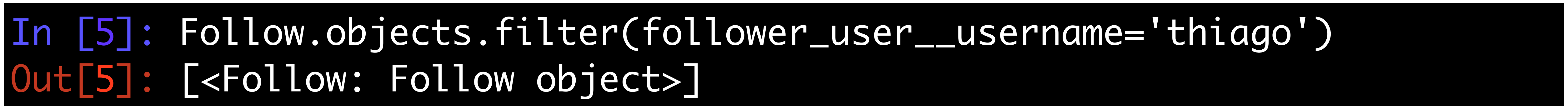
Django, Querysets

- Let’s dive into the shell



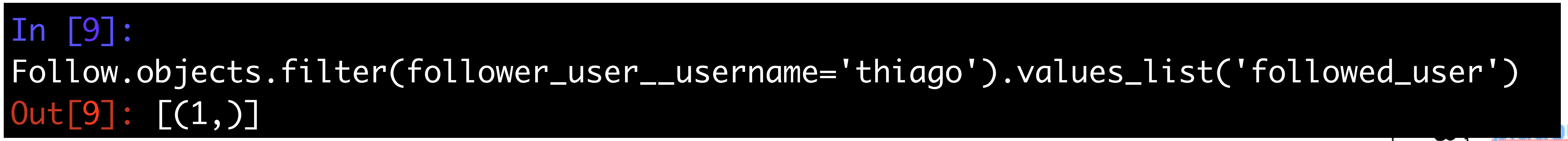
```
In [1]: from piadouro_website.models import *

In [4]: Follow.objects.filter(follower_user='thiago')
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

- wrong!!! user field expect an id (PK), is difficult to make a join and use the username inside user =P



```
In [5]: Follow.objects.filter(follower_user__username='thiago')
Out[5]: [<Follow: Follow object>]
```

- Lets get a list of ID's



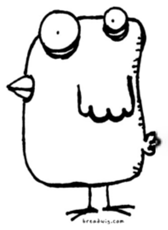
```
In [9]:
Follow.objects.filter(follower_user__username='thiago').values_list('followed_user')
Out[9]: [(1,)]
```

Django, Querysets

- So what we can do with this list

```
In [9]:
Follow.objects.filter(follower_user__username='thiago').values_list('followed_user')
Out[9]: [(1,)]

In [10]: followed_ids =
Follow.objects.filter(follower_user__username='thiago').values_list('followed_user')

In [11]: Piado.objects.filter(user__in=followed_ids)
Out[11]: [<Piado: Piado object>]
```

Magic!



piadouro

Django, Querysets

- TODO
  - user Querysets to show only tweets of followed users into home screen
  - Figure out how to use values() function with Piado object an set the home template to show links to user in every tweet (google it!)
  - Remeber to add a /newpiado link to our tool, so the users can add new Piado!
  - What about try to enable multiple users (share the tool with your colleagues)

Final Repository

- Fonte:
  - http://bit.ly/1wAb6g5
- Repo do curso
  - http://bit.ly/1sk1Rex
- Contato:
  - gabriel@elabsis.com
  - twiter: @escovabr
  - fb.com/gdcavalcante
  - fb.com/elaborainfo