Introduction to Django

Master in Free Software 2012, July 14 2012 Joaquim Rocha <jrocha@igalia.com>

What is it?

"Django is a highlevel Python Web framework that encourages rapid development and clean, pragmatic design."

(from Django's official webpage)

What is it?

Created by Lawrence Journal-World en 2003

Should help journalists meet hard deadlines

Should not stand in the way of journalists

Named after the famous Jazz guitar player Django Reinhardt

The framework

Object-Relational Mapper (ORM)

Automatic admin interface

Elegant URL design

Powerful templating system

i18n

Big community

Django has a big community and an extense list of apps

Search for them in http://github.com, http://code.google.com or http://djangopackages.com

Other interesting web pages:

Django Planet: https://www.planetdjango.org

Django Community: https://www.djangoproject.com/community

Django Sites: http://www.djangosites.org

Django People: http://www.djangopeople.org

Production Environment

```
Nginx + Gunicorn
mod_wsgi
FastCGI
mod_python
```

. . .

DB Backend

Officially supported:

PostreSQL MySQL SQLite Oracle

Using Django

Development Environment

Download it and install Django from http://djangoproject.com/download or

Be smart and use Python's VirtualEnv

VirtualEnv

A self-contained virtual environment for Python development

Advantages:

- * Does not touch your Python installation
- * Keep track of needed modules with a requirements file
- * Allows to test several package versions

Install VirtualEnv and Django

```
Install VirtualEnv:
# apt-get install python-virtualenv

Create a virtual environment:
$ virtualenv my_virtual_env
```

Use a virtual environment:

```
$ cd my_virtual_env
$ source bin/activate
```

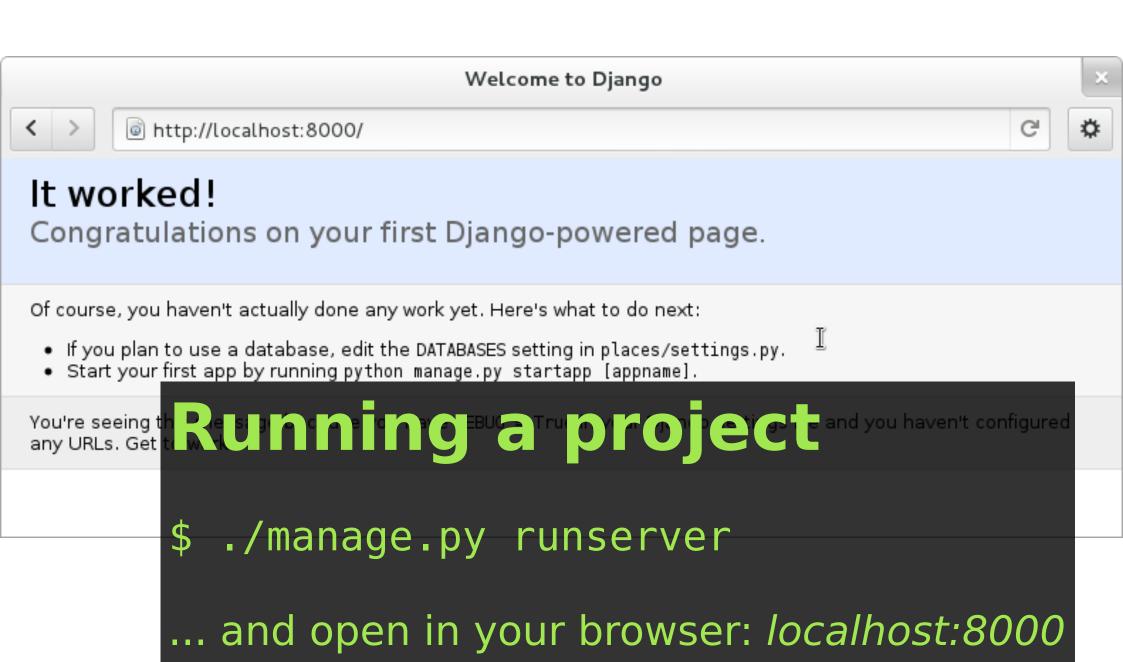
Install Django inside your virtual environment:
\$ pip install django

Development

Creating a project

\$ django-admin.py startproject myproject

```
myproject/
manage.py
myproject/
myproject/
__init__.py
settings.py
urls.py
wsgi.py
```



Development

Django projects are composed by apps

apps are the projects' modules

Creating an application

```
Inside a project, do:
$ python manage.py startapp my_app
my app/
```

init_.py
__init__.py
models.py
tests.py
views.py

Build the DB

\$ python manage.py syncdb

Project configuration

Easy configuration in file settings.py

Development

Django follows the MTV design pattern

Model-Template-View

Models

Models are classes that represent objects in the database

And you'll never have to touch SQL again!

Models

```
class Post(models.Model):
    title = models.CharField(max_length = 500)
    content = models.TextField()
    date = models.DateTimeField(auto_now = True)
    ...
```

Views

Views are functions that usually process models and render HTML

It's where the magic happens!

How to get all posts from the last 5 days and order them by descending date?

View

Templates

They will help you not repeating yourself!

And designers won't have to touch code:

URLs

In Django, URLs are part of the design!

urls.py use regular expressions to map URLs with views

URLs

Class-based Views

Allow to structure views and reuse code. They are also faster to use, for common tasks like listing or showing objects:

```
from django.views.generic import DetailView, ListView
urlpatterns = patterns('Project.posts.views',
  (r'^view/(?P<pk>d+)/$', DetailView.as_view(model=Post)),
  (r'^posts/$', ListView.as_view(model=Post)),
```

By default they try to use the templates: post_detail.html and post_list.html but these can be overriden

Classes that represent HTML forms

They allow to easily configure the expected type of inputs, error messages, labels, etc.

```
def create post(request):
    if request.method == 'POST':
        form = CreatePost(request.POST)
    if form.is valid():
        # Create a new post object with data
        # from form.cleaned data
        return HttpResponseRedirect('/index/')
    else:
        form = CreatePost()
        return render to response('create.html', {
                                   'form': form,
```

Forms for Models

Forms that are created automatically for modelos, just like:

```
from django.forms import models

class PostForm(models.ModelForm):
    class Meta:
        model = Post
```

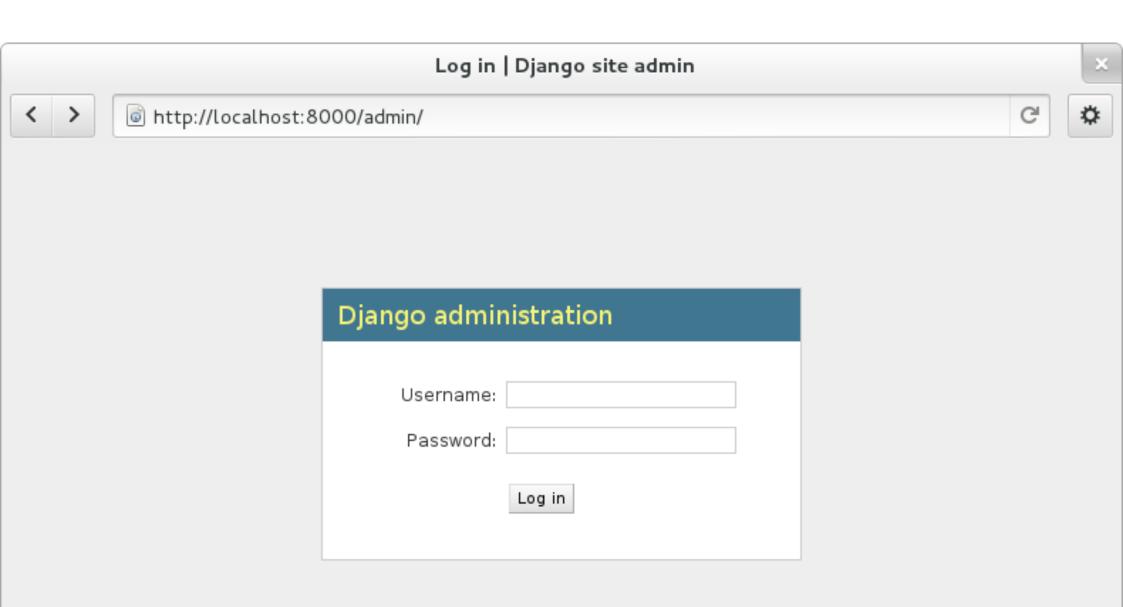
Automatic Admin

To enable automatic admin, follow the instructions in your project's URLs (uncomment the admin URLs) and uncomment the admin app in settings

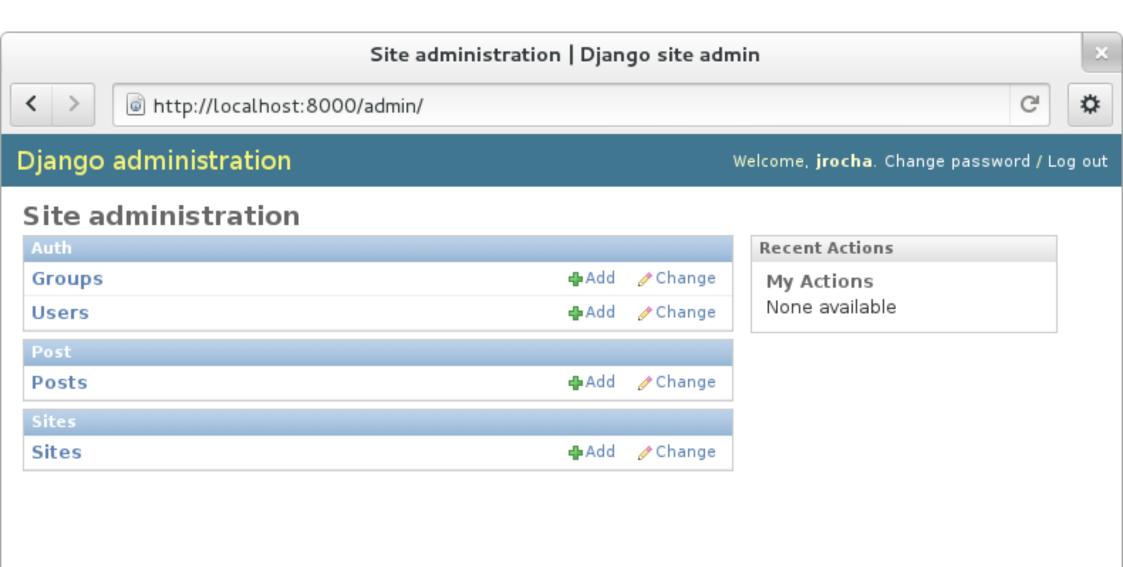
Then, to add a model to the admin, create an *admin.py* file inside an app, for example, for the Post model:

```
from django.contrib import admin from models import Post
```

admin.site.register(Post)



This gives you automatic login...



... and creation, edition, deletion of objects

Next steps

Django friendly hosts

An extense list can be found at:

http://code.djangoproject.com/wiki/DjangoFriendlyWebHosts

Google AppEngine also makes it easy for Django projects to be deployed:

http://appengine.google.com/

Help

Django Documentation:

https://docs.djangoproject.com

Cheat sheet:

http://www.revsys.com/django/cheatsheet/

Some books:

The Django Book: http://www.djangobook.com/ Learning Website Development with Django, Packt Practical Django Projects, Apress Pro Django, Apress