

Django

01 Introdução

Professor: Ritomar Torquato

01 Introdução

Objetivos: conhecer interfaces de linha de comando; O instalador de pacotes; Ambientes Virtuais; Instalação do Django; Alô Mundo! E mais...

Referências



<https://docs.djangoproject.com/>

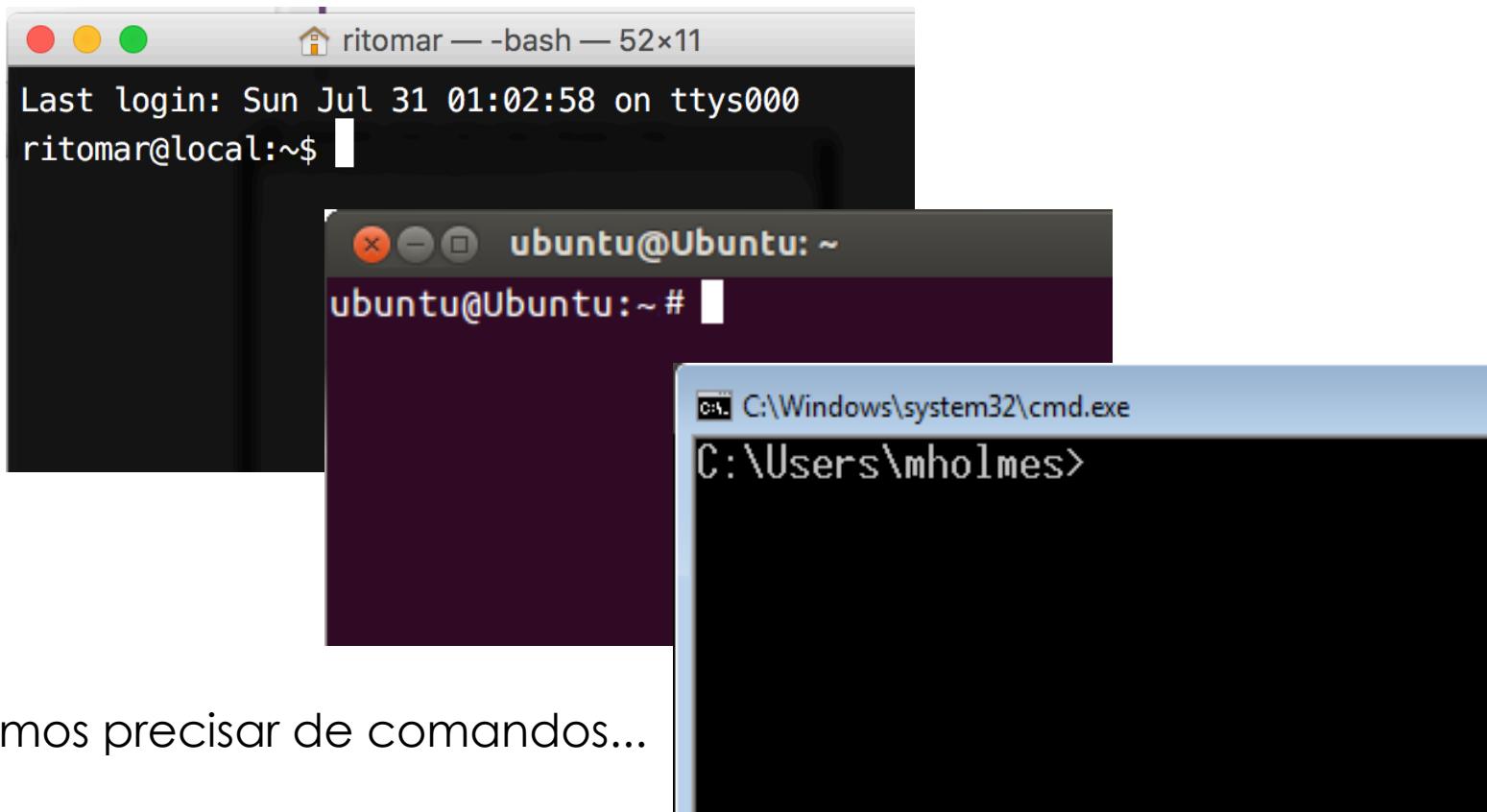


<http://tutorial.djangogirls.org/pt/>



<http://www.udemy.com/python-3-na-web-com-django-basico-intermediario/>

Linha de Comandos



Linha de Comandos

Não se assuste, é só um programa esperando comandos.



O terminal é um programa baseado em **texto** para visualização, manipulação ou execução de arquivos em seu computador (como no Windows Explorer, mas sem interface gráfica).

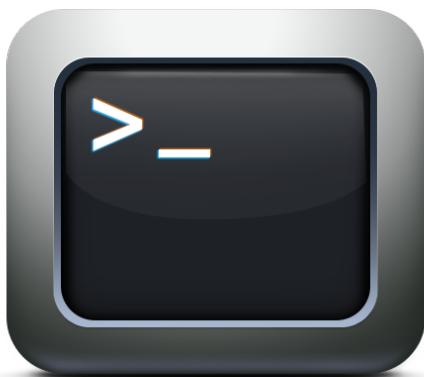
Nomes comuns são:

cmd, prompt, console, shell ou terminal.

Django precisa muito do terminal de comandos.

Linha de Comandos

Não se assuste, é só um programa esperando comandos.



Praticamente tudo que é possível fazer pelo Windows Explorer, ou no equivalente em Linux ou MAC OS, é possível fazer pelo terminal usando linha de comando:
criar arquivos, renomear, executar programas, criar pastas, mover-se pelas pastas, etc.

Que comandos são esses?
Veremos já... Antes precisamos abrir o terminal.

Linha de Comandos

Abrir o prompt de comandos.

Windows

Iniciar → Todos os Programas → Acessórios → Prompt de comando

Mac OS X

Applications → Utilities → Terminal

Linux

Applications → Accessories → Terminal

Depende muito, qualquer coisa é só procurar no Google.

Linha de Comandos

Abrir o prompt de comandos.



No Windows, o padrão é aparecer um sinal de >

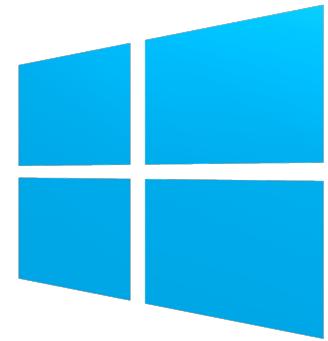
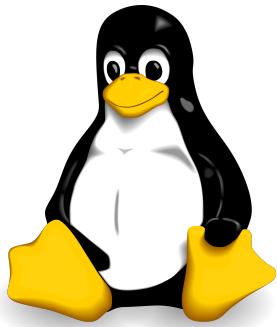


Se você estiver em Mac ou num Linux, você provavelmente verá um \$

Isso pode mudar, não tem problema.

Linha de Comandos

Comandos podem ou não variar pelo Sistema Operacional.



Digite o **whoami** seguido de ENTER (todos os sistemas)

```
C:\Users\Ritomar Torquato>whoami  
vbox-win\ritomar torquato
```

```
ritomar@local:~$ whoami  
ritomar
```

mostra seu nome de usuário.

Linha de Comandos

Algumas convenções

Usaremos a formatação abaixo para indicar os comandos que devem ser executados no terminal:

```
> whoami      # Isso é comentário e deve ser ignorado.  
computador\ritomar torquato
```

```
$ whoami      # No MAC isso também deve ser ignorado.  
ritomar torquato
```

O caractere ">" indica Windows e "\$" Linux ou MAC. Sem um desses caracteres o comando ser para qualquer sistema.

Após digitar um comando é necessário pressionar a tecla **ENTER** para que sua execução seja feita.

Linha de Comandos

Mostrar a pasta (ou diretório) atual

```
> cd          # change directory  
C:\Users\ritomar
```

```
$ pwd          # print working directory  
/Users/ritomar
```

A pasta de início padrão é a home do usuário atual.

No Linux/MAC as pastas são representadas por barra normal (/),
no Windows são representadas por barra invertida (\)

Diretório é outro nome para pasta. Mais adiante usaremos cd para trocar de pasta.

Linha de Comandos

Listando arquivos e pastas

```
> dir
```

```
$ ls
```

Mostra uma listagem com todos os arquivos e pastas que existem na pasta atual.

Nota: 'dir' é uma abreviação para '**directory**' e 'ls' é uma abreviação para '**list**'.

Linha de Comandos

Entrar em (ou sair de) pastas ou diretórios

```
cd <caminho desejado>
```

```
cd Desktop      # Entra na pasta "Desktop"
```

```
cd ..          # Volta para pasta de nível superior
```

```
cd "My docs"   # Se o nome contiver espaço use aspas
```

```
$ cd ~         # Vai para a pasta Home do usuário atual (~)
```

```
$ cd /         # Vai para a pasta raiz do sistema (/)
```

```
> cd \        # Vai para a pasta raiz do disco atual (\)
```

Linha de Comandos

Entrar em (ou sair de) pastas ou diretórios

```
ritomar@local:~$ ls
Applications      Downloads      Keys          Music
Projects          Themes         Desktop       Dropbox
Library           OneDrive      Public        Documents
GitHub            Movies

ritomar@local:~$ cd D          # Pressione o TAB duas vezes
Desktop/          Documents/   Downloads/  Dropbox/
```

Windows, Linux e MAC: Usar "cd <string>" (por exemplo "cd D") e apertar a tecla TAB, o sistema tenta preencher automaticamente o resto do nome para que você possa navegar rapidamente. Se houver várias opções, aperte a tecla TAB novamente.

Linha de Comandos

Entrar em (ou sair de) pastas ou diretórios

```
ritomar@local:~/Desktop$
```

```
C:\Users\Ritomar_Torquato\Desktop>
```

É comum o prompt de comandos exibir a pasta de que estamos no momento, antes do ">" ou do "\$".

Você sempre pode recorrer ao comando "> cd" ou "\$ pwd" para exibir em qual pasta você está atualmente.

Linha de Comandos

Criando uma pasta

```
mkdir Nome_da_nova_pasta
```

```
mkdir "Nome da nova pasta"
```

```
mkdir ifpi
```

```
mkdir "Instituto Federal do Piauí"
```

mkdir (make directory) criar a pasta com o nome informado.
Verifique se foi criada com "\$ ls" ou "> dir"!

DICA: Use seta para cima e seta para baixo no teclado para percorrer comandos usados recentemente.

Linha de Comandos

Removendo arquivos e pastas

Atenção: A exclusão é irrecuperável, tenha cuidado com estes comando.

```
$ rm -rf ifpi
```

```
> rmdir /S ifpi  
ifpi, Tem certeza <S/N>? S
```

Verifique se foi removida com "\$ ls" ou "> dir"!

Linha de Comandos

Fechando o terminal...

```
exit
```

OBSERVAÇÃO IMPORTANTE:

O Windows não diferencia caracteres maiúsculos de minúsculos, assim exit, Exit e EXIT é o mesmo comando;

Já no Linux/MAC só é válido exit (tudo minúsculo).

Essa observação é válida para qualquer comando visto.

Linha de Comandos

Resumindo os principais comandos!

Comando (Windows)	Comando (Mac/Linux)	Descrição
exit	exit	Fechá a janela
exit		
cd	cd	Muda a pasta
cd destino		
dir	ls	Lista as pastas e os arquivos
> dir		
\$ ls		

Linha de Comandos

Resumindo os principais comandos!

Comando (Windows)	Comando (Mac/Linux)	Descrição
copy	cp	Copia um arquivo
> copy origem.py destino.py > copy c:\test\test.txt c:\windows\test.txt \$ cp ~/test/test.txt ~/Desktop/test.txt		
move	mv	Move um arquivo
> move origem.py destino.py > move c:\test\test.txt c:\windows\test.txt \$ mv ~/test/test.txt ~/Desktop/test.txt		

Linha de Comandos

Resumindo os principais comandos!

Comando (Windows)	Comando (Mac/Linux)	Descrição
mkdir	mkdir	Cria uma pasta
mkdir testdirectory		
del	rm	Apaga um arquivo
> del c:\meutexto.txt		
\$ rm ~/meutexto.txt		
rmdir	rm	Apaga uma pasta
> rmdir pasta		

Linha de Comandos

Comandos próprios do Python e do Django.

AGUARDE!

Prática

Pratique com os comandos baixo:

```
$ mkdir ifpi  
$ cd ifpi  
$ mkdir teste  
$ ls  
teste
```

```
> mkdir ifpi  
> cd ifpi  
> mkdir teste  
> dir
```

05/08/2014 07:28 PM <DIR>

teste

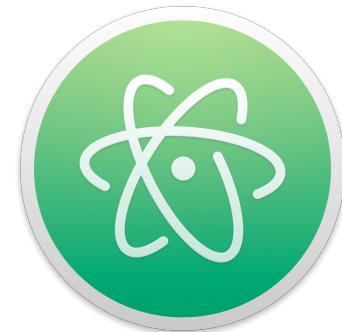
Editor de código

Gedit

wiki.gnome.org/Apps/Gedit



sublime
www.sublimetext.com/



atom
atom.io/

Python Installing Packages (pip)

Instalador de Pacotes Python - pip

Página Oficial

<https://pip.pypa.io/>

Preciso Instalar o pip?

O pip já deve está instalado se você usa Python 2 >= 2.7.9 ou Python 3 >= 3.4 obtido do site oficial (python.org), talvez você precisar atualizar.

Tenho o pip?

```
pip -V          # Senão funcionar tente pip2 ou pip3
pip 7.1.2 from ...
```

Python Installing Packages (pip)

Instalador de Pacotes Python - pip

Atualizar o pip?

```
$ pip install -U pip
```

```
> python -m pip install -U pip
```

Python Installing Packages (pip)

Instalador de Pacotes Python - pip

Instalar o pip?

Se estiver usando Debian/Ubuntu:

```
$ sudo apt-get install python-pip
```

Ou você pode fazer o download do arquivo
<https://bootstrap.pypa.io/get-pip.py> e executar o seguinte comando:

```
python get-pip.py # Se for o caso, use python3
```

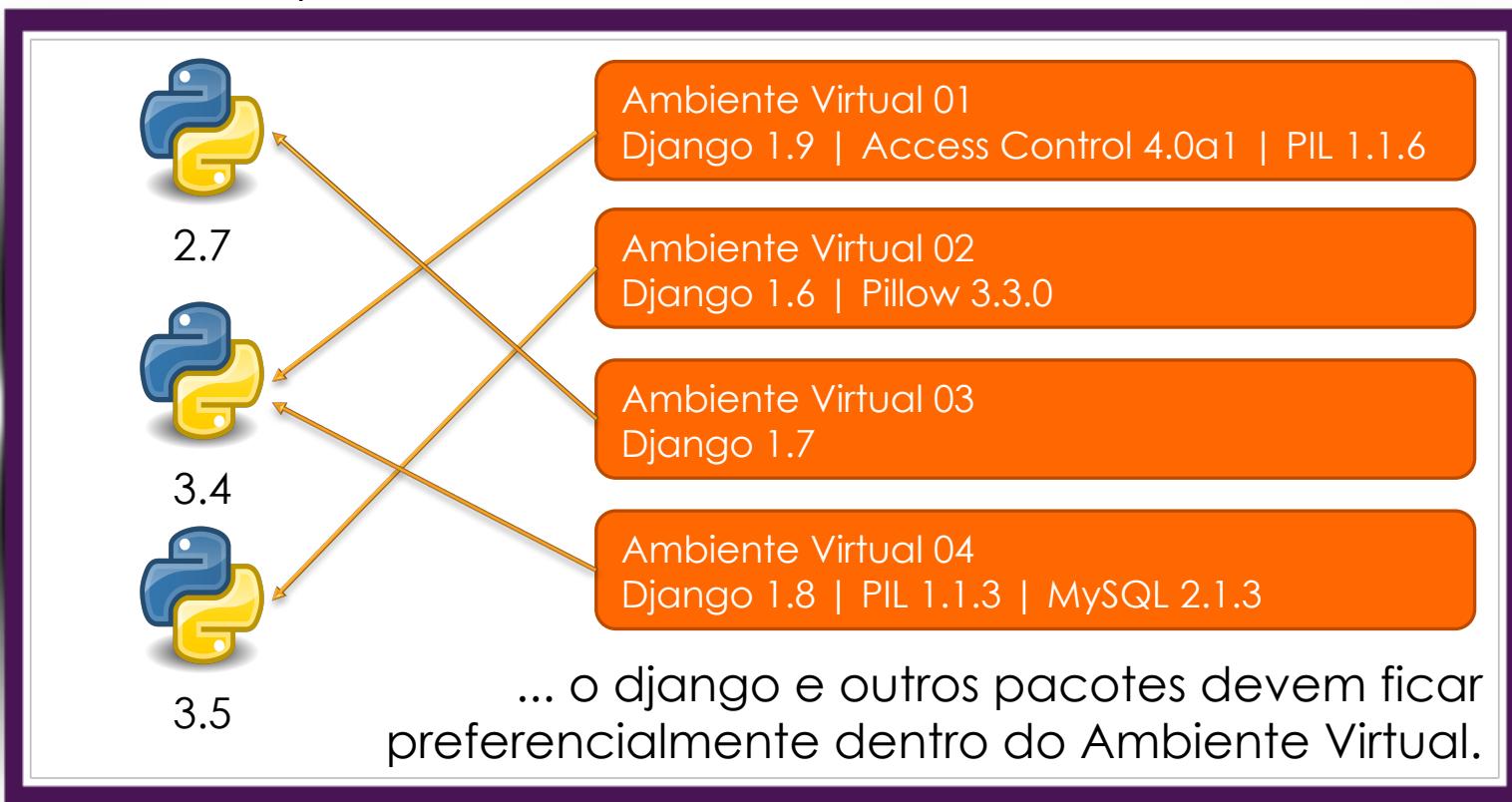
Ambiente Virtual

Com vários projetos em andamento, é bastante comum que um projeto tenha dependências de bibliotecas diferentes das usadas pelos outros projetos.

Podemos criar um ambiente virtual e instalar os pacotes necessários dentro desse ambiente, sem a necessidade de ter privilégios de super-usuário e eles não vão interferir em outros projetos.

Ambiente Virtual

No seu computador...



Ambiente Virtual

Usando o Ambiente Virtual

A forma comum de criar ambientes virtuais é com o [virtualenv](#). Verifique se ele já está instalado com seguinte comando:

```
virtualenv --version
```

Caso não possua, você pode instalar o VirtualEnv através do gerenciador de pacotes pip. Tendo o pip instalado, use o seguinte comando no Linux/MAC ou Windows:

```
$ sudo pip install virtualenv
```

```
> pip install virtualenv
```

Ambiente Virtual

Usando o Ambiente Virtual

Agora podemos criar os ambientes virtuais para projetos Python:

```
virtualenv myenv
```

O comando acima cria um ambiente e utiliza o python padrão do Sistema Operacional, se deseja um outro pode fazer:

```
virtualenv -p python3 myenv
```

Quando criamos o ambiente, é criado um diretório com o nome informado (myenv), com as seguintes pastas:

bin/ include/ lib/ local/

O virtualenv gerencia esse conteúdo para manter o ambiente separado. Remover essa pasta é remover o ambiente virtual.

Ambiente Virtual

Usando o Ambiente Virtual

Após o ambiente ter sido criado, precisamos ativá-lo:

```
$ source ./myenv/bin/activate  
(myenv) $
```

```
> myenv\Scripts\activate  
(myenv) >
```

O (nome do ambiente) entre parênteses indica que ele está ativado e pronto para uso. Agora podemos instalar pacotes no ambiente virtual ativado que não vão interferir no sistema.

Instalando o Django

O Django é o primeiro pacote!

Agora que temos uma virtualenv ativada, vamos instalar Django usando pip.

```
(myenv) pip install django
```

O comando acima instala a versão mais recente do Django no ambiente virtual que está ativo (myenv)

Muitas vezes é preferível indicar a versão desejada, podemos fazer isso da forma mostrada abaixo:

```
(myenv) pip install django==1.8.5
```

Perceba que usamos um duplo sinal de igual (==).

Instalando o Django

O Django é o primeiro pacote!

Para verificar se tudo correu bem durante a instalação, execute os comandos em destaque a seguir:

```
(myenv) $ python
Python 3.5.1
>>> import django
>>> django.get_version()
'1.9.8'
```

De forma reduzida...

```
(myenv) $ python -m django -version
1.9.8
```

Instalando o Django

O Django é o primeiro pacote!

no Windows

Se você receber um erro ao chamar o pip, verifique se o caminho do projeto contém espaços, acentos ou caracteres especiais (exemplo, C:\Users\João\ifpi). Se sim, prefira outro lugar sem espaços, acentos ou caracteres especiais (sugestão é: C:\ifpi). Após a mudança, por favor tente novamente o comando.

No Linux

Se você receber um erro ao chamar pip no Ubuntu 12.04 por favor execute `python -m pip install -U --force-reinstall pip` para corrigir a instalação do pip no virtualenv.

Criando um projeto

Vamos começar um novo projeto de Django. Isto significa que executar alguns scripts fornecidos pelo Django que irá criar o esqueleto de um projeto: pastas e arquivos que usaremos.

```
(myenv) $ django-admin startproject mysite
```

```
(myenv) > django-admin startproject mysite
```

O comando inicia em django-admin, (myenv) indica apenas que estamos com o virtualenv ativo e o caractere seguinte qual sistema operacional usamos.

Django-admin cria uma pasta com nome mysite que é a pasta do nosso projeto.

Criando um projeto

Pastas e arquivos criados por django-admin:

```
mysite/
└── manage.py
└── mysite/
    ├── settings.py
    ├── urls.py
    ├── wsgi.py
    └── __init__.py
```

manage.py é um script que ajuda com a gestão do projeto.
settings.py contém a configuração do projeto.
urls.py contém uma lista dos caminhos (urls) do projeto.
Por enquanto vamos ignorar os outros arquivos.

Criando um projeto

Executando o servidor de desenvolvimento

Na pasta que contém o arquivo manage.py (o diretório mysite) podemos iniciar o servidor web com o comando:

```
(aula) $ python manage.py runserver
```

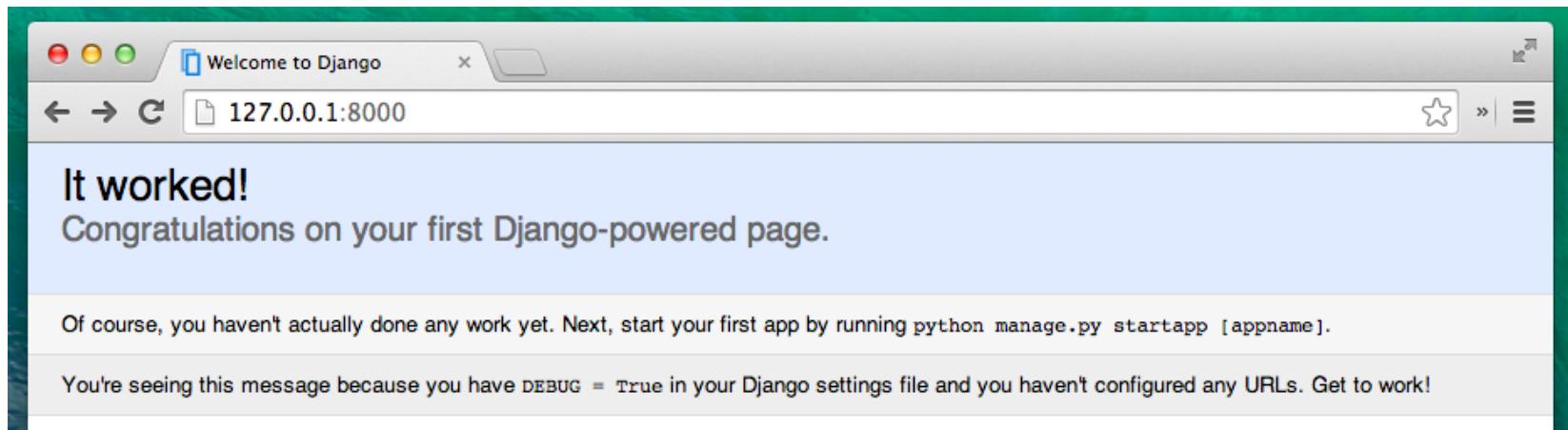
```
Performing system checks...
Django version 1.9.8,
Starting development server at
http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Parabéns! Você criou e executou seu primeiro site django em um servidor de web, vamos vê-lo...

Criando um projeto

Executando o servidor de desenvolvimento

Abra seu navegador no endereço 127.0.0.1:8000 ...



Criando um projeto

Alterando as configurações do projeto

Vamos alterar o `mysite/settings.py` para nosso horário e idioma.

Localize as linha que contém `LANGUAGE_CODE` e `TIME_ZONE` e modifique para nosso idioma e fuso horário, respectivamente. Veja como deve ficar:

```
# Internationalization
# https://docs.djangoproject.com/en/1.9/topics/i18n/
```

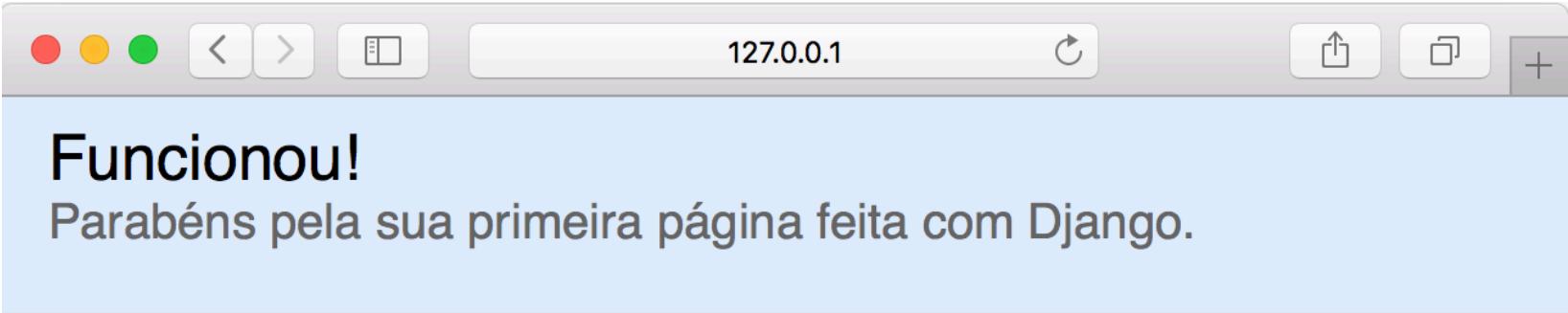
```
LANGUAGE_CODE = 'pt-br'
```

```
TIME_ZONE = 'America/Fortaleza'
```

Criando um projeto

Executando o servidor de desenvolvimento

Recarregue sua página para ver o resultado...



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1`. The main content area displays the text "Funcionou! Parabéns pela sua primeira página feita com Django." Below this, a note says: "É claro que você realmente não fez qualquer trabalho ainda. O próximo passo é iniciar o seu primeiro app rodando `python manage.py startapp [app_label]`." At the bottom, another note states: "Você está vendo esta mensagem, porque você tem `DEBUG = True` no seu arquivo de configurações do Django e você não configurou nenhum URLs. Vamos ao trabalho!"

Funcionou!
Parabéns pela sua primeira página feita com Django.

É claro que você realmente não fez qualquer trabalho ainda. O próximo passo é iniciar o seu primeiro app rodando `python manage.py startapp [app_label]`.

Você está vendo esta mensagem, porque você tem `DEBUG = True` no seu arquivo de configurações do Django e você não configurou nenhum URLs. Vamos ao trabalho!

Criando uma aplicação

Django usa aplicações para manter o código organizado. Um projeto é um conjunto de aplicações. Estamos prontos para criar nossa primeira aplicação.

Estando na mesma pasta que está o arquivo `manage.py`, execute o comando:

```
(myenv) $ python manage.py startapp core
```

Core será o nome do nosso primeiro aplicativo. É uma boa prática de programação em Django usar um nome genérico neste momento.

Um novo diretório `core` é criado com alguns arquivos.

Criando uma aplicação

Em nossa pasta mysite uma nova pasta core foi criada como este:

```
core/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

Criando uma aplicação

Precisamos informar ao nosso projeto que uma nova aplicação foi criada e que ele deve usá-la. Fazemos isso no settings.py

Em INSTALLED_APPS adicione uma linha com 'core'. O resultado será semelhante a esse:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
  
    'core',  
]
```

Escrevendo a primeira view

Uma view é um gerenciador responsável pelo recebimento das requisições e geração da resposta adequada.

Abra o arquivo /core/views.py e coloque o código Python a seguir:

```
# /mysite/core/views.py

from django.http import HttpResponse

def index(request):
    return HttpResponse("Alô, Mundo!")
```

A view recebe um request que é uma requisição http e retorna um resposta HttpResponse.

Configurando uma rota

Rotas são os caminhos disponíveis em nosso site a partir da url base.
Cada view deve responder à uma rota.
Definimos as rotas no arquivo urls.py

```
from django.conf.urls import url
from django.contrib import admin

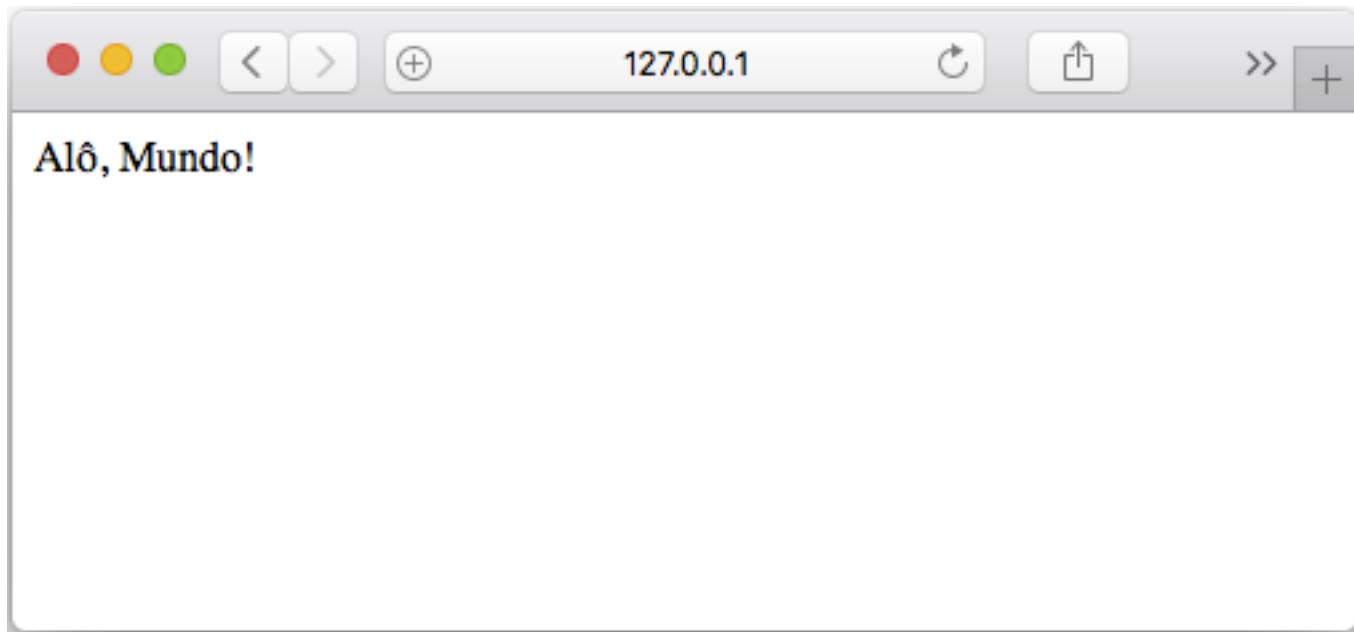
→ from core import views

urlpatterns = [
    → url(r'^$', views.index, name='index'),
    url(r'^admin/', admin.site.urls),
]
```

Ao receber uma requisição o django tenta localizar a view responsável pelo tratamento no arquivo urls.py.

Tudo pronto!

Já podemos visualizar nossa aplicação rodando. Não esqueça de executar o servidor de desenvolvimento do Django.



Que sem graça!



Mantando a sua fome

Indo do Alô Mundo à uma agenda de contatos com persistência de dados e autenticação de usuário em 4 passos.

Como
Assim?



Mantando a sua fome

1. Criar um modelo
 - Editar core/models.py
2. Registrar o modelo no administrador
 - Editar /core/admin.py
3. Gerar o banco de dados
 - python manage.py makemigrations
 - python manage.py migrate
4. Criar um super-usuário
 - python manage.py createsuperuser

Mantando a sua fome

1. Criar um modelo

```
# /mysite/core/models.py

from django.db import models

class Contato(models.Model):
    nome = models.CharField(max_length=50)
    endereco = models.CharField(max_length=200)
    email = models.EmailField(max_length=100)
    data_nascimento = models.DateField()
    telefone = models.CharField(max_length=20)

    def __str__(self):
        return self.nome
```

Mantando a sua fome

2. Registrar o modelo no administrador

```
# /mysite/core/admin.py

from django.contrib import admin

from .models import Contato

class ContatoAdmin(admin.ModelAdmin):
    list_display = ('nome', 'email', 'telefone')

admin.site.register(Contato, ContatoAdmin)
```

Mantando a sua fome

3. Gerar o banco de dados

- python manage.py makemigrations
- python manage.py migrate

```
python manage.py makemigrations
```

```
python manage.py migrate
```

Mantando a sua fome

4. Criar um super-usuário

- `python manage.py createsuperuser`

```
python manage.py createsuperuser
```

```
Username (leave blank to use 'myuser'):
```

```
Email address: myuser@mysite.com
```

```
Password:
```

```
Password (again):
```

```
Superuser created successfully.
```

Tudo pronto!

Vamos acessar o site em **http://127.0.0.1/admin**
Não esqueça de executar o servidor de desenvolvimento do Django.

Administração do Django

Usuário:

Senha:

Acessar

Use o usuário e senha criados para acessar.