**Curso – DJANGO**

1. Criando uma virtual machine

C:\Users\Name\djangogirls> python -m venv myvenv

1. Ativar/entrar na virtual machine

C:\Users\Name\djangogirls> myvenv\Scripts\activate

1. Sair da Virtual machine

Deactivate

1. Criando um projeto Django
   1. django-admin startproject myproject

**Configuração do ambiente DJANGO**

**Mudando as configurações**

Vamos fazer algumas alterações no mysite/settings.py. Abra o arquivo usando o editor de código que você instalou anteriormente.

**Observação:** Lembre-se de que o settings.py é um arquivo comum, como qualquer outro. Você pode abri-lo de dentro do editor de código usando as ações de menu "Arquivo-> Abrir". Assim, você deve encontrá-lo na janela usual para selecionar arquivos e abri-lo. Ou então, é possível abrir o arquivo navegando até o diretório do djangogirls e abrindo o arquivo com o botão direito. Uma vez clicado, selecione o seu editor de código preferido da lista. Selecionar o editor apropriado é importante uma vez que você pode ter outros programas instalados que podem abrir o arquivo, mas não editá-lo.

Para começar, seria bom ter a hora correta no nosso site. Para isto, você configurar o fuso horário correto de onde está. Se você estiver no Brasil, é bem provável que o fuso horário seja America/Sao\_Paulo (aqui conhecido como horário de Brasília). Caso queira saber mais, vá para [Wikipedia's list of time zones](https://en.wikipedia.org/wiki/List_of_tz_database_time_zones" \t "_blank) e copie e cole o fuso horário correspondende à sua localização.

Em settings.py, localize a linha que contém TIME\_ZONE e modifique para escolher seu próprio fuso horário:

mysite/settings.py

TIME\_ZONE = 'America/Sao\_Paulo'

Um código de idioma se refere à língua, por exemplo, en para inglês ou pt para português e o código do país, por exemplo, br para Brasil ou pt para a Portugal. Já que o inglês provavelmente não é sua língua nativa, você pode pode adicionar um novo código de país para deixar os botões padrão e notificações de Django em seu idioma. Assim, você teria por exemplo um botão "Cancel" traduzido para a língua da sua escolha (ex: "Cancelar" em português). O [Django vem com um monte de traduções já preparadas](https://docs.djangoproject.com/en/2.0/ref/settings/" \l "language-code" \t "_blank).

Se você quiser um idioma diferente do inglês, especifique o código de idioma alterando a seguinte linha:

mysite/settings.py

LANGUAGE\_CODE = 'pt-BR'

Também precisamos adicionar o caminho para os arquivos estáticos. (Discutiremos tudo sobre arquivos estáticos e CSS mais adiante no tutorial.) Vá até o *final* do arquivo e, logo abaixo da linha com STATIC\_URL, adicione uma nova variável chamada STATIC\_ROOT:

mysite/settings.py

STATIC\_URL = '/static/'

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')

Quando DEBUG for True e ALLOWED\_HOSTS estiver vazia, o domínio do site será validado como ['localhost', '127.0.0.1', '[::1]']. Isso não corresponderá ao nosso domínio no PythonAnywhere quando implantarmos a nossa aplicação, então vamos mudamos a seguinte configuração:

mysite/settings.py

ALLOWED\_HOSTS = ['127.0.0.1', '.pythonanywhere.com']

**Observação:** Se você estiver utilizando um Chromebook, adicione esta linha ao final do arquivo settings.py: MESSAGE\_STORAGE = 'django.contrib.messages.storage.session.SessionStorage'

Também inclua .c9users.io à lista de ALLOWED\_HOSTS se você estiver utilizando o cloud9

**Configurando um banco de dados**

Existem vários software de banco de dados diferentes que podem armazenar dados para o seu site. Nós vamos usar o padrão do Django, o sqlite3.

Isto já está configurado nesta parte do seu arquivo mysite/settings.py:

mysite/settings.py

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

Para criar um banco de dados para o nosso blog, vamos executar o seguinte comando no console. Digite: python manage.py migrate (precisamos estar no diretório que contém o arquivo manage.py djangogirls). Se isso der certo, você deve ver algo assim:

command-line

(myvenv) ~/djangogirls$ python manage.py migrate

Operations to perform:

Apply all migrations: auth, admin, contenttypes, sessions

Running migrations:

Rendering model states... DONE

Applying contenttypes.0001\_initial... OK

Applying auth.0001\_initial... OK

Applying admin.0001\_initial... OK

Applying admin.0002\_logentry\_remove\_auto\_add... OK

Applying contenttypes.0002\_remove\_content\_type\_name... OK

Applying auth.0002\_alter\_permission\_name\_max\_length... OK

Applying auth.0003\_alter\_user\_email\_max\_length... OK

Applying auth.0004\_alter\_user\_username\_opts... OK

Applying auth.0005\_alter\_user\_last\_login\_null... OK

Applying auth.0006\_require\_contenttypes\_0002... OK

Applying auth.0007\_alter\_validators\_add\_error\_messages... OK

Applying auth.0008\_alter\_user\_username\_max\_length... OK

Applying auth.0009\_alter\_user\_last\_name\_max\_length... OK

Applying sessions.0001\_initial... OK

Pronto! Hora de iniciar o servidor web e ver se nosso site está funcionando!

**Iniciando o servidor web**

Você precisa estar no diretório que contém o arquivo manage.py (o diretório djangogirls). No console, nós podemos iniciar o servidor web executando o python manage.py runserver:

command-line

(myvenv) ~/djangogirls$ python manage.py runserver