Django

Documentação

Escrevendo seu primeiro app Django, parte 1

Vamos aprender por um exemplo.

Através deste tutorial, nós vamos caminhar através da criação de uma aplicação básica de enquetes.

Ela irá consistir de duas partes:

- Uma página pública que permitirá que pessoas vejam as enquetes e votem nelas.
- Um site de administração que permite adicionar, alterar e deletar enquetes.

Assumiremos que você já tem o Django instalado. Você pode verificar se o Django está instalado e em qual versão rodando o seguinte comando em um prompt shell (indicado pelo prefixo $):

```
$ python -m django --version
```

Se o Django estiver instalado, você verá a versão de sua instalação. Se não estiver, você receberá uma mensagem de erro dizendo “No module named django”.

Este tutorial foi escrito para Django 3.2, com suporte para Python 3.6 ou mais atual. Se a versão do Django não é a mesma, você pode acessar o tutorial da sua versão de Django usando o menu de versões no canto inferior desta página, ou atualizando o Django para uma versão mais recente. Se estiver usando uma versão mais antiga de Python, veja Qual versão do Python eu posso usar com Django? para encontrar uma versão compatível do Django.

Veja Como instalar o Django para recomendação sobre como remover versões antigas do Django e instalar uma mais recente.

Onde obter ajuda:

Se tiver problemas enquanto caminha por este tutorial, por favor consulte a seção Obtendo ajuda da FAQ.

Criando um projeto

Se esta é a primeira vez em que você está utilizando o Django, você terá que preocupar-se com algumas configurações iniciais. Isto é, você precisará auto-gerar um código para iniciar um projeto Django – uma coleção de configurações para uma instância do Django, incluindo configuração do banco de dados, opções específicas do Django e configurações específicas da aplicação.

A partir da linha de comando, execute **`cd`** para o diretório onde você gostaria de armazenar seu código, então execute o seguinte comando:

```
$ django-admin startproject mysite
```

Isto irá criar o diretório **`mysite`** no seu diretório atual. Se não funcionar, veja Problemas ao rodar o django-admin.

Nota

Você precisará evitar dar nomes a projetos que remetam a componentes internos do Python ou do Django. Particularmente, isso significa que você deve evitar usar nomes como **`django`** (que irá conflitar com o próprio Django) ou **`test`** (que irá conflitar com um pacote interno do Python).

Onde esse código deveria existir?

Se você tem experiência prévia em PHP (sem o uso de um framework moderno), você deve estar acostumado a colocar o código dentro do “document root” de seu servidor Web (em um lugar como **`/var/www`**). Com o Django você não fará isto. Não é uma boa ideia colocar qualquer código Python no document root de seu servidor Web, porque existe o risco de pessoas conseguirem ver seu código através da Web. Isso não é bom para a segurança.

Coloque seu código em algum diretório **fora** do document root, como em **`/home/mycode`**.

Vamos ver o que o **`startproject`** criou:

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

Esses arquivos são:

- O diretório **`mysite/`** exterior é um contêiner para o seu projeto. Seu nome não importa para o Django; você pode renomeá-lo para qualquer nome que você quiser.
- **`manage.py`**: um utilitário de linha de comando que permite a você interagir com esse projeto Django de várias maneiras. Você pode ler todos os detalhes sobre o **`manage.py`** em django-admin and manage.py.
- O diretório **`mysite/`** interior é o pacote Python para o seu projeto. Seu nome é o nome do pacote Python que você vai precisar usar para importar coisas do seu interior (por exemplo, **`mysite.urls`**).
- **`mysite/__init__.py`**: um arquivo vazio que diz ao Python que este diretório deve ser considerado um pacote Python. Se você é um iniciante Python, leia mais sobre pacotes na documentação oficial do Python.
- **`mysite/settings.py`**: configurações para este projeto Django. Configurações do Django irá revelar para você tudo sobre o funcionamento do **`settings`**.
- **`mysite/urls.py`**: as declarações de URLs para este projeto Django; um "índice" de seu site movido a Django. Você pode ler mais sobre URLs em Despachante de URL.
- **`mysite/asgi.py`**: um ponto de integração para servidores web compatíveis com ASGI usado para servir seu projeto. Veja Como fazer o deploy com ASGI para mais detalhes.
- **`mysite/wsgi.py`**: um ponto de integração para servidores web compatíveis com WSGI usado para servir seu projeto. Veja Como implementar com WSGI para mais detalhes.

O servidor de desenvolvimento

Vamos verificar se ele funciona. Vá para o diretório **`mysite`**, se você ainda não estiver nele, e execute o seguinte comando:

```
$ python manage.py runserver
```

Você verá a seguinte saída na sua linha de comando:

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.

julho 09, 2021 - 15:50:53
Django version 3.2, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Nota

Por hora, ignore os avisos sobre as migrações do banco de dados não aplicadas; lidaremos com o banco de dados brevemente.

Você iniciou o servidor de desenvolvimento do Django, um servidor Web leve escrito puramente em Python. Nós incluímos ele com o Django, então você pode desenvolver coisas rapidamente, sem ter que lidar com a configuração de um servidor Web de produção – como o Apache – até que você esteja pronto para a produção.

Agora é uma boa hora para notar: **não** use esse servidor em nada relacionado a um ambiente de produção. Seu objetivo é ser utilizado apenas durante o desenvolvimento. (Nós estamos na atividade de criação de frameworks Web, não de servidores Web.)

Agora que o servidor está rodando, visite http://127.0.0.1:8000/ no seu navegador Web. Você verá uma página de "Parabéns!", com um foguete decolando. Funciona!

Alterando a porta

Por padrão, o comando **`runserver`** inicia o servidor de desenvolvimento no IP interno na porta 8000.

Se você quer mudar a porta do servidor, passe ela como um argumento na linha de comando. Por exemplo, este comando iniciará o servidor na porta 8080:

```
$ python manage.py runserver 8080
```

Se você quer mudar o IP do servidor, passe junto com a porta. Por exemplo, para escutar em todos os IPs públicos disponívies (o que é útil se estiver usando Vagrant ou quer mostrar seu trabalho para outros computadores na rede), use:

```
$ python manage.py runserver 0:8000
```

0 é um atalho para **0.0.0.0**. A documentação completa para o servidor de desenvolvimento pode ser encontrada na referência do **`runserver`** .

Recarregamento automática de `runserver`

O servidor de desenvolvimento recarrega o código Python para cada solicitação, conforme necessário. Você não precisa reiniciar o servidor para que alterações de código tenham efeito. No entanto, algumas ações como a adição de arquivos não disparam uma reinicialização, então você terá que reiniciar o servidor nestes casos.

Criando a aplicação de enquetes: Polls

Agora que seu ambiente – um "projeto" – está configurado, você está pronto para começar o trabalho.

Cada aplicação que você escreve no Django consiste de um pacote Python que segue uma certa convenção. O Django vem com um utilitário que gera automaticamente a estrutura básica de diretório de uma aplicação, então você pode se concentrar apenas em escrever código em vez de ficar criando diretórios.

Projetos versus aplicações

Qual é a diferença entre um projeto e uma aplicação? Uma aplicação é um conjunto de elementos web que faz alguma coisa - por exemplo, um sistema de blog, um banco de dados de registros públicos, ou uma pequena aplicação de enquetes. Um projeto é uma coleção de configurações e aplicações para um website particular. Um projeto pode conter múltiplas aplicações. Uma aplicação pode estar em múltiplos projetos. Em Django, chamamos uma aplicação de "app".

Suas aplicações, ou "apps", podem ficar em qualquer lugar dentro do seu caminho de busca de módulos Python. Neste tutorial, iremos criar nossa "app" de enquete no mesmo diretório que seu arquivo **`manage.py`** para que possa ser importado como um módulo de nível superior dentro do projeto, ao invés de ser um submódulo de **`mysite`**.

Para criar sua aplicação, certifique-se de que esteja no mesmo diretório que **`manage.py`** e digite o seguinte comando:

```
$ python manage.py startapp polls
```

Que irá criar o diretório **`polls`** com a seguinte estrutura:

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


Esta estrutura de diretório irá abrigar a aplicação de enquete.

Escreva sua primeira view

Hora de criar a primeira view. Abra o arquivo **`polls/views.py`** e ponha o seguinte código em Python dentro dele:

polls/views.py

```
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

Esta é a view mais simples possível no Django. Para chamar a view, nós temos que mapear a URL - e para isto nós precisamos de uma URLconf.

Para criar uma URLconf no diretório **`polls`**, crie um arquivo chamado **`urls.py`**. Agora seu diretório da aplicação deve ficar como:

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    urls.py
    views.py
```

No arquivo **`polls/urls.py`** inclua o seguinte código:

polls/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

O próximo passo é apontar na raiz da URLconf para o módulo **`polls.urls`**. No arquivo **`mysite/urls.py`**, adicione uma importação de **`django.urls.include`** e insira um **`include()`** na lista **`urlpatterns`**, de forma que você tenha:

mysite/urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

A função **`include()`** permite referenciar outras URLconfs. Qualquer lugar que o Django encontrar **`include()`**, irá recortar todas as partes da URL encontrada até aquele ponto e enviar a string restante para URLconf incluído para processamento posterior.

A idéia por trás do **`include()`** é facilitar plugar URLs. Uma vez que polls está em sua própria URLconf (**`polls/urls.py`**), ele pode ser colocado depois de "/polls/", ou depois de "/fun_polls/", u depois de "/content/polls/", ou qualquer outro início de caminho, e a aplicação ainda irá funcionar

Quando usar `include()`

Deve-se sempre usar **`include()`** quando você incluir outros padrões de URL. **`admin.site.urls`** é a única exceção a isso.

Agora você amarrou uma view **`index`** no seu URLconf. Verique se funciona com o seguinte comando:

```
$ python manage.py runserver
```

Acesse http://localhost:8000/polls/ no seu navegador, e deverá ver o texto "*Hello, world. You're at the polls index.*", o qual foi definido na view "index".

Página não encontrada?

Se retornar uma página de erro, verifique se está acessando http://localhost:8000/polls/ e não http://localhost:8000/.

A função **`path()`** são passado quatro argumentos, dois obrigatórios: **`route`** e **`view`**, e dois opcionais: **`kwargs`**, e **`name`**. Neste ponto, vale a pena revisar para o que estes argumentos servem.

Argumento de `path()`: `route`

`route` é uma string contento uma descrição de uma URL. Quando processa uma requisição, o Django começa pela primeira descrição, e vai descendo a lista, comparando a URL requisitada com cada descrição até que encontre uma que combine.

Descrições não distinguem paramentros GET e POST, ou o nome do domínio. Por exemplo, em uma requisição **`https://www.example.com/myapp/`**, o URLconf irá procurar por **`myapp/`**. Em uma requisição para **`https://www.example.com/myapp/?page=3`, o URLconf também irá procurar por ``myapp/`**.

Argumento de `path()`: `view`

Quando o Django encontra uma descrição que combina, chama a função view especificada com um objeto **`HttpRequest`** como primeiro argumento e qualquer valor "capturado" da rota como argumentos keyword. Daremos um exemplo sobre isso logo.

Argumento de `path()`: `kwargs`

Argumentos nomeados arbitrários podem ser passadas em um dicionário para a view de destino. Nós não vamos usar este recurso do Django neste tutorial.

Argumento de `path()`: `name`

Nomear sua URL permite que você referencie ela de forma inequívoca de qualquer lugar no Django especialmente nos templates. Este poderoso recurso permite que você faça alterações globais nos padrões de URL de seu projeto enquanto modifica um único arquivo.

Quando estiver confortável com o básico da sequência de requisição e resposta, leia a parte 2 desse tutorial para começar a trabalhar com banco de dados.

‹ **Guia de instalação rápida**

Escrevendo sua primeira aplicação Django, parte 2 ›

Diversity Statement

Get Involved

Join a Group

Contribute to Django

Submit a Bug

Report a Security Issue

Follow Us

GitHub

Twitter

News RSS

Django Users Mailing List

Support Us

Sponsor Django

Official merchandise store

Amazon Smile

Benevity Workplace Giving Program