

## Aula 02

### Django

#### urls

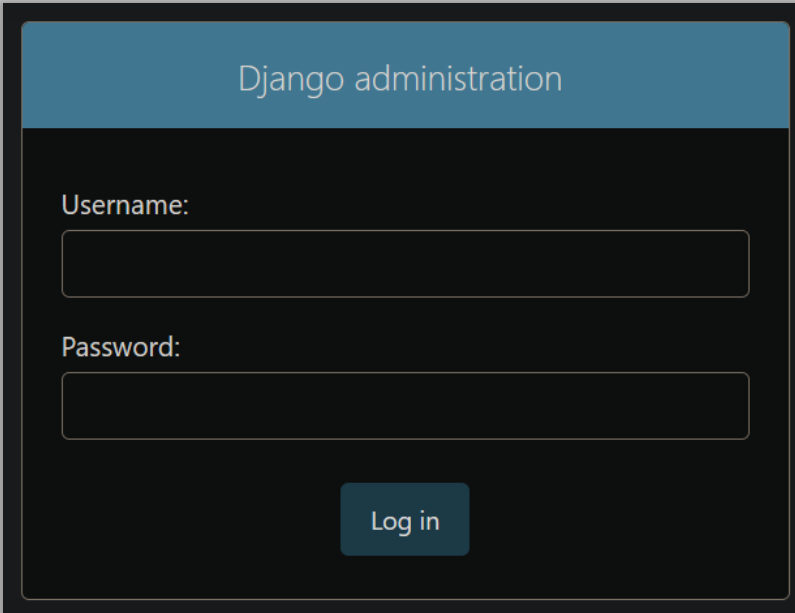
O arquivo **urls.py** é a porta de entrada do nosso site, do nosso projeto.

Quando acessamos qualquer site em Django, estamos acessando primeiro esse arquivo. Ele nos permite configurar esse comportamento.

Por enquanto, essa é a configuração do nosso arquivo urls.py:

```
C:\Users\gutoh\curso\curso\urls.py
22 """
21 URL configuration for curso project.
20
19 The 'urlpatterns' list routes URLs to views. For more information please see:
18 ...https://docs.djangoproject.com/en/4.2/topics/http/urls/
17 Examples:
16 Function views
15 ...1. Add an import: from my_app import views
14 ...2. Add a URL to urlpatterns: path('', views.home, name='home')
13 Class-based views
12 ...1. Add an import: from other_app.views import Home
11 ...2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
10 Including another URLconf
9 ...1. Import the include() function: from django.urls import include, path
8 ...2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
7 """
6 from django.contrib import admin
5 from django.urls import path
4
3 urlpatterns = [
2     path('admin/', admin.site.urls),
1 ]
23
```

Nesse arquivo, a variável `urlpatterns` vai especificar os subdomínios do nosso site. Se observarmos, nosso projeto já tem um subdomínio, o `admin`. Podemos acessar indo no browser e digitando <http://127.0.0.1:8000/admin>, que vai nos levar para a página abaixo:



Django administration

Username:

Password:

Log in

Essa é a tela de login para a página administrativa do nosso projeto. Por enquanto nenhum usuário está configurado, então não temos como entrar. Veremos isso mais adiante.

Voltando ao `urlpatterns`, sabendo como funciona, podemos adicionar alguns subdomínios (que a partir de agora chamaremos de aplicativos) personalizados lá seguindo o seguinte critério:

- usar a função `path`;
- passar para ela como primeiro parâmetro uma string com o nome do aplicativo;
- e uma função que recebe um parâmetro e retorne um objeto do tipo `HttpResponse`;

Vamos criar um aplicativo manualmente, por enquanto.

Dica: podemos colocar o mouse a função `path` para vermos a documentação dela.

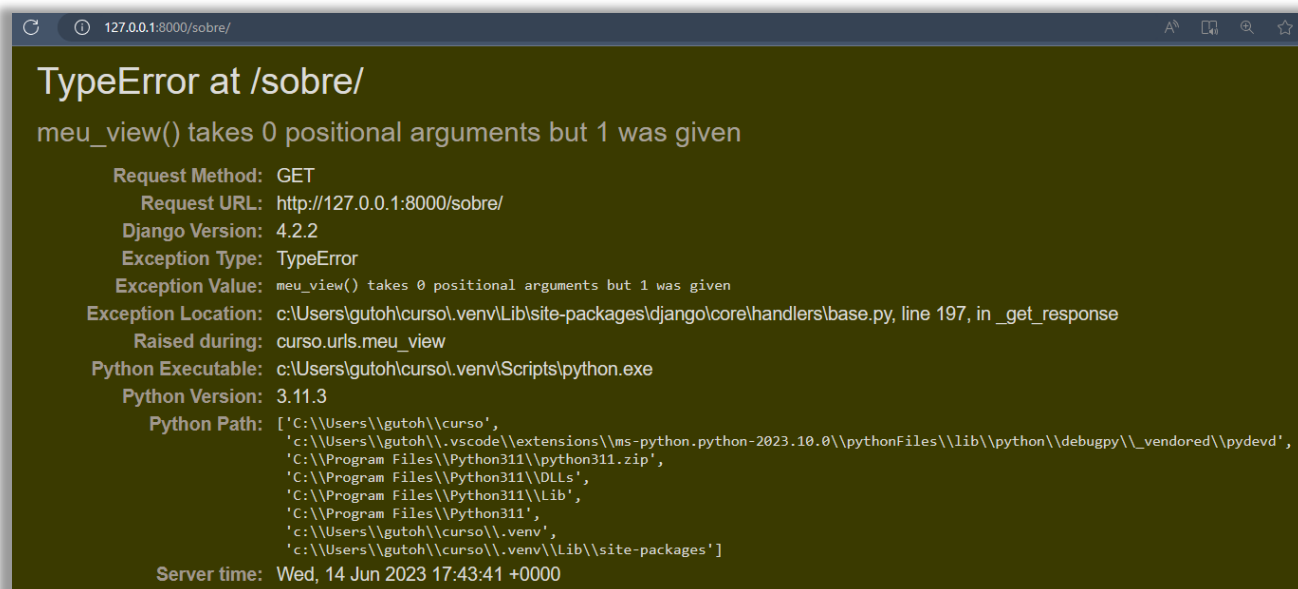
Veja como vai ficar o arquivo com a função que criamos:

```
C:\Users\gutoh\curso\curso\urls.py
10 from django.contrib import admin
9  from django.urls import path
8
7  def meu_view():
6      ...
5
4  urlpatterns = [
3      ...path('admin/', admin.site.urls),
2      ...path('sobre/', meu_view)
1  ]
11
```

PS.: é o mesmo arquivo da primeira imagem, apenas removi o comentário para poupar espaço (veja o caminho absoluto do arquivo na primeira linha).

Agora, vamos executar o servidor e acessar o subdomínio criado através do endereço <http://127.0.0.1:8000/sobre>.

Então, veremos a seguinte mensagem de erro:



Veja o erro apresentado acima (acostume-se às mensagens de erro, elas são fundamentais para entender o problema e o funcionamento do projeto), foi passada à função `meu_view` um argumento, mas ela não esperava nenhum.

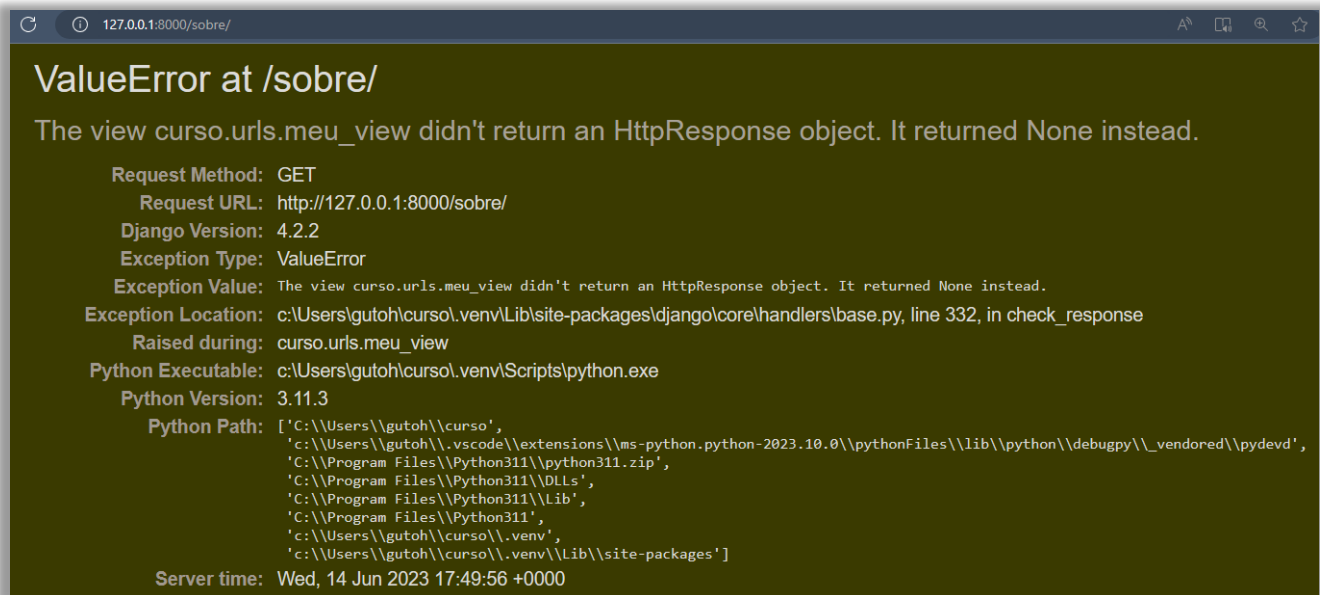
Esse argumento passado é um request, que é uma requisição que realizamos pelas urls. Ele é recebido pelo servidor e retornado como uma página web.

É dessa forma que funciona um servidor web. O cliente faz um HTTP Request e o servidor responde com um HTTP Response.

Então, vamos incluir esse parâmetro na nossa função:

```
C:\Users\gutoh\curso\curso\urls.py
10 from django.contrib import admin
9  from django.urls import path
8
7  def meu_view(request):
6      ...
5
4  urlpatterns = [
3      ...path('admin/', admin.site.urls),
2      ...path('sobre/', meu_view)
1  ]
11
```

Se recarregarmos a página, veremos outro erro agora:



Agora, falta nossa função retornar um objeto do tipo HttpResponse. Como não especificamos o que retornar, é retornado None. Lembre-se, é esperado uma resposta http da nossa página.

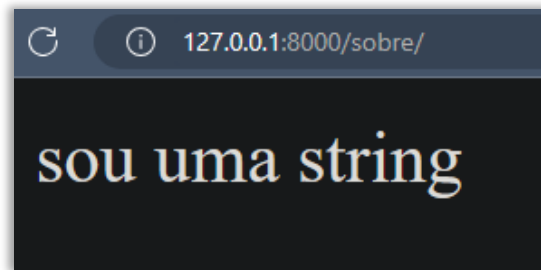
Então, vamos adicionar esse objeto no retorno da nossa função.

Para isso, temos que importar a Classe HttpResponse (conforme visto abaixo). Essa classe espera receber uma string como argumento para criação do objeto.

Veja como vai ficar:

```
C:\Users\gutoh\curso\curso\urls.py
11 from django.contrib import admin
10 from django.urls import path
9  from django.http import HttpResponse
8
7  def meu_view(request):
6      ...return HttpResponse('sou uma string')
5
4  urlpatterns = [
3      ...path('admin/', admin.site.urls),
2      ...path('sobre/', meu_view)
1  ]
12
```

Se recarregarmos nossa página no endereço <https://127.0.0.1:8000/sobre/>, teremos nossa primeira página web personalizada.



Alguns pontos importantes:

1. repare que nossa página inicial desapareceu (aquela com a apresentação do Django);
2. a string passada para o path não pode ter uma barra antes, mas tem que ter uma no final;
3. essa string retornada na função pode ser formatada como um HTML;

Para praticar, vamos criar mais alguns aplicativos para nosso projeto:

```
C:\Users\gutoh\curso\curso\urls.py
1 from django.contrib import admin
2 from django.urls import path
3 from django.http import HttpResponse
4
5 def view_home(request):
6     return HttpResponse('<h1>sou a home</h1>')
7
8 def view_sobre(request):
9     return HttpResponse('<h2>sou a sobre</h2>')
10
11 def view_contato(request):
12     return HttpResponse('<p>sou o contato</p>')
13
14 urlpatterns = [
15     path('admin/', admin.site.urls),
16     path('', view_home), # recriando a home
17     path('sobre/', view_sobre), # novo nome
18     path('contato/', view_contato), # nova página
19 ]
20
```

Agora, nosso projeto já tem 4 páginas web, que podemos navegar digitando o nome de cada página.

Experimente acessar os links <https://127.0.0.1:8000/>, <https://127.0.0.1:8000/sobre/> e <https://127.0.0.1:8000/contato/>.

## Aplicativos

Repare como fica trabalhoso criar funções para cada url e, geralmente, essas funções serão grandes blocos de códigos.

Por isso, o Django nos permite criar aplicativos (que representam os subdomínios do nosso site). Então, para cada página, teremos aplicativos individuais. Isso também nos permite realizar a portabilidade deles para outros sites apenas copiando a referida pasta, sem ter que recriar tudo. Assim, alteramos apenas o necessário.

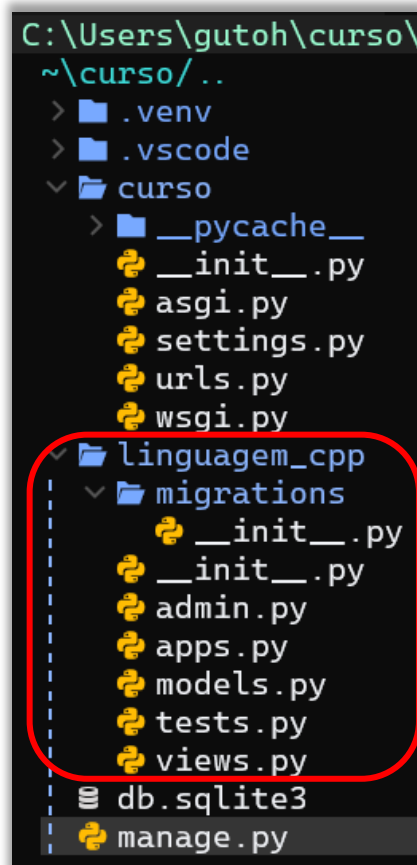
Vamos criar nosso primeiro aplicativo.

Vamos chamar ele de `linguagem_cpp`, que será sobre a linguagem de programação C++. Para isso, temos que interromper nosso servidor e executar:

```
(.venv) C:\Users\gutoh\curso>python manage.py startapp linguagem_cpp
```

**IMPORTANTE:** todos os aplicativos criados precisam ser uma única palavra, sem o uso de espaços ou qualquer outro caractere especial diferente do sublinhado (como o usado acima).

Observe o que foi criado agora:



Repare que temos uma nova pasta (irmã da pasta `curso`) com o nome que especificamos no comando (`linguagem_cpp`). Dentro dela, há diversos módulos python. Temos o `__init__.py`, que faz com que a pasta seja reconhecida como um módulo do python.

A pasta **migrations** serve para configurarmos nosso banco de dados, mas não vamos nos preocupar com isso agora.

O que importa são os novos arquivos:

- **admin.py** e **models.py**: servem para o mapeamento objeto-relacional (ORM), que é usado para o banco de dados do aplicativo;
- **apps.py**: nome do seu aplicativo criado, quando alteramos o **settings.py**, vamos usar o valor da variável name ou o caminho completo; veremos mais sobre isso adiante;

- **tests.py**: é usado para testarmos nossa aplicação (não todo o site, mas apenas esse aplicativo);
- **views.py**: arquivo que será usado para criarmos nossas páginas, será onde colocaremos as funções, como as criadas anteriormente;

Uma vez criado o aplicativo, agora podemos mover todas as funções criadas no arquivo /curso/urls.py para o /linguagem\_cpp/views.py.

Veja como ele vai ficar:

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
13 from django.http import HttpResponse
12 from django.shortcuts import render
11
10 # Create your views here.
9
8 def view_home(request):
7     ...return HttpResponse('<h1>sou a home do curso C++</h1>')
6
5 def view_sobre(request):
4     ...return HttpResponse('<h2>sou a sobre do curso C++</h2>')
3
2 def view_contato(request):
1     ...return HttpResponse('<p>sou o contato do curso C++</p>')
14
```

(lembre-se que também temos que mover a importação do HttpResponse para esse módulo).

Uma vez removidas as funções de /curso/urls.py, temos que adicionar a referência delas da nova localização.

Veja como vai ficar o arquivo /curso/urls.py agora:

```
C:\Users\gutoh\curso\curso\urls.py
11 from django.contrib import admin
10 from django.urls import path
9 from linguagem_cpp.views import view_home, view_sobre, view_contato
8
7
6 urlpatterns = [
5     ...path('admin/', admin.site.urls),
4     ...path('', view_home), # recriando a home
3     ...path('sobre/', view_sobre), # atualizando o que criamos
2     ...path('contato/', view_contato) # nova página
1 ]
12
```

Agora podemos navegar nas páginas, como estávamos fazendo antes.

## Exercícios para Praticar

1. Se ainda estiver inseguro quanto aos passos feitos até agora, execute novamente o passo a passo da aula 01 e 02 de uma só vez.
2. Se já estiver seguro quando à aula 01, repita os passos apenas dessa aula.
3. Crie mais aplicativos para seu projeto e adicione diferentes páginas a cada um.
4. Crie um código em html dentro de uma variável string e passe ela para o `HttpResponse`.
5. Crie um arquivo do tipo html na pasta de algum aplicativo criado, abra e leia ele com a função `open()` para uma variável e passe essa variável para o `HttpResponse`.