Aula 13

Django

Carregando Dados

Quando salvamos os dados no banco de dados usando a página de administração, o Django automaticamente tem acesso a eles. Para acessar esses dados, usamos um gerenciador específico do Django.

Para acessar esses dados, temos que chamar o arquivo **/linguagem_cpp/models.py** no arquivo **/linguagem_cpp/views.py** e então usar a classe que criamos para chamar as características herdadas da classe Model:

C:\Users\gutoh\curso\linguagem_cpp\views.py

```python
from django.shortcuts import render
from utils.versoes.fabrica import cria_versao
from . import models

VERSOES = [cria_versao() for _ in range(5)]

def view_inicio(request):
    dicionario = {'nome': 'Tom', 'sobrenome': 'Cruise'}
    return render(request, 'linguagem_cpp/paginas/inicio.html', context=dicionario)

def view_sobre(request):
    return render(request, 'linguagem_cpp/paginas/sobre.html')

def view_contato(request):
    return render(request, 'linguagem_cpp/paginas/contato.html')

def view_documentacao(request):
    dicionario = {'vers': VERSOES}
    versoes = models.Versao.objects.all()
    print(versoes)

    return render(request, 'linguagem_cpp/paginas/documentacao.html', context=dicionario)

def view_versao(request, id):
    resultado = {}
    for item in VERSOES:
        if item['versao'] == id:
            resultado = item
            break
    return render(request, 'linguagem_cpp/paginas/versao.html', context=resultado)
```

Quando carregarmos a página início, vamos ver o seguinte no terminal:

```
(.venv) C:\Users\gutoh\curso>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 06, 2023 - 16:31:41
Django version 4.2.2, using settings 'curso.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

<QuerySet [<Versao: Versão 1>, <Versao: Versão 2>]>
[06/Jul/2023 16:31:45] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Isso nos mostra que estamos carregando os dados do banco de dados.

O que está sendo mostrado é uma QuerySet. Ele é um objeto do Django que fica responsável por buscar os dados no banco de dados e retornar para a aplicação.

Comandos

O principal comando para buscar todos os dados de uma tabela é o all():

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.all()
20     print(versoes)
```

Que, como mencionado anteriormente, vai retornar uma lista do conteúdo da tabela:

```
<QuerySet [<Versao: Versão 1>, <Versao: Versão 2>]>
[06/Jul/2023 16:31:45] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Há também formas de retornar os resultados, como ordenar por algum atributo da classe (que também é o nome do campo na tabela):

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.all()
20     print(f'versoes → {versoes}')
21     print(f'versoes.order_by("titulo") → {versoes.order_by("titulo")}')
22     print(f'versoes.order_by("-titulo") → {versoes.order_by("-titulo")}')
```

O método .order_by() é usado para ordenarmos os resultados buscados no banco de dados. O padrão é ordenar pela chave primária em ordem crescente.

Abaixo, temos a busca ordenando pelo título usando a ordem ascendente e a ordem descendente, que é representada pelo símbolo – (sinal de menos) junto do nome do atributo:

```
versoes → <QuerySet [<Versao: Versão 1>, <Versao: Versão 2>]>
versoes.order_by("titulo") → <QuerySet [<Versao: Versão 1>, <Versao: Versão 2>]>
versoes.order_by("-titulo") → <QuerySet [<Versao: Versão 2>, <Versao: Versão 1>]>
[06/Jul/2023 16:43:59] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Também podemos combinar mais de um campo de ordenação:

```
C:\Users\gutoh\curso\linguagem_cpp\views.py [+]
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.all()
20     print(f'versoes → {versoes}')
21     print(f'versoes.order_by("titulo,id") → {versoes.order_by("titulo,id")}')
```

Acima, estamos ordenando primeiro pelo título em ordem crescente e, se houver mais de um registro com o mesmo nome, é ordenado pelo id em ordem crescente.

Como o que está sendo retornado uma lista de objetos do tipo Versao, podemos iterar por ela usando o loop for:

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.all()
20
21     for versao in versoes:
22         print(versao)
23         print(versao.__dict__)
```

Que vai nos retornar:

```
Versão 1
{'_state': <django.db.models.base.ModelState object at 0x000001DC77373350>, 'id': 2, 'titulo':
 'Versão 1', 'versao': 1, 'slug': 'ver-1', 'descricao': 'Primeira Versão do C++', 'data_lanc':
 datetime.date(2023, 7, 4), 'eh_publicado': True, 'foto': 'imagens/2023/07/06/frances.jpg'}
Versão 2
{'_state': <django.db.models.base.ModelState object at 0x000001DC77372AD0>, 'id': 3, 'titulo':
 'Versão 2', 'versao': 2, 'slug': 'ver-2', 'descricao': 'Segunda versão do C++', 'data_lanc':
datetime.date(2023, 7, 5), 'eh_publicado': False, 'foto': 'imagens/2023/07/06/foto-perfil.jpg'
}
[06/Jul/2023 17:14:59] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Se tratando de um objeto, acessamos seus valores do atributos usando a notação do ponto entre o objeto e seu atributo.

Também podemos usar o método .first() para retornar apenas o primeiro elemento daquela busca:

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.all()
20
21     print(f'versoes.first() → {versoes.first()}')
```

```
versoes.first() → Versão 1
[06/Jul/2023 17:19:44] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

O elemento retornado, vai vir na forma do objeto.

Se quiser, podemos realizar o ordenamento da lista usando o order_by() e usar o – para deixar a lista em ordem reversa e então usar o first() para recuperar o último elemento da lista.

Há uma forma de buscar por todos os campos e tipos do objeto buscado no banco de dados. Usando o _meta.get_field() é nos retornado uma tupla com todos os dados:

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.all()
20
21     print(f'versoes[0]._meta.get_fields() → {versoes[0]._meta.get_fields()}')
```

```
versoes[0]._meta.get_fields() → (<django.db.models.fields.BigAutoField: id>, <django.db.models.fields.CharField: ti
tulo>, <django.db.models.fields.IntegerField: versao>, <django.db.models.fields.SlugField: slug>, <django.db.models.
fields.CharField: descricao>, <django.db.models.fields.DateField: data_lanc>, <django.db.models.fields.BooleanField:
 eh_publicado>, <django.db.models.fields.files.ImageField: foto>)
[06/Jul/2023 17:25:05] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Depois de buscados os dados no banco de dados com o método all(), podemos realizar um filtro do que foi buscado.

Por exemplo, podemos filtrar todos as versões que têm valor True no campo eh_publicado:

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.filter(eh_publicado=True)
20
21     print(f'versoes → {versoes}')
```

```
versoes → <QuerySet [<Versao: Versão 1>]>
[06/Jul/2023 17:31:07] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Há também outra forma de realizar esse filtro usando o .get():

```
C:\Users\gutoh\curso\linguagem_cpp\views.py
16
17 def view_documentacao(request):
18     dicionario = {'vers': VERSOES}
19     versoes = models.Versao.objects.get(eh_publicado=True)
20
21     print(f'versoes → {versoes}')
```

```
versoes → Versão 1
[06/Jul/2023 17:33:48] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Há uma grande diferença entre os dois métodos. O filter retorna uma lista de um item, enquanto que o get retorna o objeto em si.

Tem que ter o cuidado na hora de usar um ou outro, já que, se a busca não encontrar qualquer registro no filter, vai retornar uma lista vazia, já o get vai gerar um erro:

Usando o filter:

```
versoes → <QuerySet []>
[06/Jul/2023 17:37:50] "GET /cpp/documentacao/ HTTP/1.1" 200 1090
```

Usando o get:

```
Internal Server Error: /cpp/documentacao/
Traceback (most recent call last):
  File "C:\Users\gutoh\curso\.venv\Lib\site-packages\django\core\handlers\exception.py", line 55, in inner
    response = get_response(request)
               ^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\gutoh\curso\.venv\Lib\site-packages\django\core\handlers\base.py", line 197, in _get_response
    response = wrapped_callback(request, *callback_args, **callback_kwargs)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\gutoh\curso\linguagem_cpp\views.py", line 19, in view_documentacao
    versoes = models.Versao.objects.get(id=30)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\gutoh\curso\.venv\Lib\site-packages\django\db\models\manager.py", line 87, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\gutoh\curso\.venv\Lib\site-packages\django\db\models\query.py", line 637, in get
    raise self.model.DoesNotExist(
linguagem_cpp.models.Versao.DoesNotExist: Versao matching query does not exist.
[06/Jul/2023 17:37:08] "GET /cpp/documentacao/ HTTP/1.1" 500 76034
```

Mais sobre a QuerySet pode ser encontrado na QuerySet API reference.

Exercícios para Praticar

1. Use o que foi aprendido em aula e aplique no seu projeto.

Atividade para Entregar

1. Requisitos para a entrega final:
    a. seu projeto tem que ter, ao menos, dois aplicativos de sua escolha e um aplicativo para servir de home para todo seu projeto;
    b. ter um menu de navegação entre TODOS os aplicativos;
    c. usar a organização de arquivos nas pastas templates (HTMLs) e static (CSSs, JSs e imagens) dentro de cada aplicativo;
    d. usar a organização de arquivos nas pastas templates (HTMLs) e static (CSSs, JSs e imagens) global;
    e. uso das tags Django if, elif, else, for, include, url, static, block, extends;
    f. criação de classes com os campos, ao menos um de cada listado abaixo:
        i. charfield
        ii. integerfield
        iii. slugfield
        iv. datefield
        v. booleanfield
        vi. imagefield
    g. usar o banco de dados para guardar os dados inseridos pelo administrador;
    h. o administrador do projeto deve ter senha e usuário "***admin***";
    i. usar imagens, css e js nas páginas;

2. O seu site deve ter uma interface viável;
    a. ter css, js, imagens e um conteúdo;
    b. Dica: use as grades do bootstrap para organizar rapidamente o geral do site (não obrigatório);
        i. o uso do bootstrap não exclui o uso de CSS próprio;