INSERÇÃO DE MÚLTIPLOS DADOS NO TEMPLATE

Utilizando TemplateView com múltiplos modelos

Para exibir várias classes em um único template, a abordagem mais prática é usar uma view baseada em TemplateView e, nela, realizar consultas para todos os modelos que deseja exibir.

Vamos supor que temos três modelos: Produto, Categoria, e Fornecedor.

Modelos no models.py

python
Copiar código

```
# models.py
from django.db import models

class Produto(models.Model):
    nome = models.CharField(max_length=100)
    preco = models.DecimalField(max_digits=10, decimal_places=2)

class Categoria(models.Model):
    nome = models.CharField(max_length=100)
    descricao = models.TextField()

class Fornecedor(models.Model):
    nome = models.CharField(max_length=100)
    contato = models.CharField(max_length=100)
```

Criando a TemplateView para exibir múltiplos modelos

No arquivo views.py, crie uma view que herda de TemplateView. No método get_context_data, você pode adicionar ao contexto dados de várias classes:

python
Copiar código

```
# views.py
from django.views.generic import TemplateView
from .models import Produto, Categoria, Fornecedor

class DashboardView(TemplateView):
    template_name = 'dashboard.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['produtos'] = Produto.objects.all()
        context['categorias'] = Categoria.objects.all()
        context['fornecedores'] = Fornecedor.objects.all()
        return context
```

Configurando a URL

No arquivo urls.py, adicione uma URL para a DashboardView:

python
Copiar código

```
# urls.py
```

```
from django.urls import path
from .views import DashboardView

urlpatterns = [
    path('dashboard/', DashboardView.as_view(), name='dashboard'),
]
```

Template dashboard.html

Agora, no template, você pode exibir os dados de Produto, Categoria e Fornecedor usando o contexto fornecido pela view:

```
html
Copiar código
<!-- dashboard.html -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Dashboard</title>
</head>
<body>
    <h1>Produtos</h1>
    <ul>
        {% for produto in produtos %}
            <li>{{ produto.nome }} - R$ {{ produto.preco }}</li>
        {% endfor %}
    </ul>

    <h1>Categorias</h1>
    <ul>
        {% for categoria in categorias %}
            <li>{{ categoria.nome }} - {{ categoria.descricao }}</li>
        {% endfor %}
    </ul>

    <h1>Fornecedores</h1>
    <ul>
        {% for fornecedor in fornecedores %}
            <li>{{ fornecedor.nome }} - Contato: {{ fornecedor.contato }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

2. Exibindo Dados de Classes em Views Separadas (Opcional)

Se precisar exibir dados de diferentes classes em templates separados e incluí-los em um template principal, você pode criar uma ListView para cada classe e, no template principal, incluir esses templates utilizando o Django {% include %}.

Aqui está como:

Criando ListView para cada Classe

```
python
Copiar código
# views.py
```

```
from django.views.generic import ListView
from .models import Produto, Categoria, Fornecedor

class ProdutoListView(ListView):
    model = Produto
    template_name = 'partials/produto_list.html'
    context_object_name = 'produtos'

class CategoriaListView(ListView):
    model = Categoria
    template_name = 'partials/categoria_list.html'
    context_object_name = 'categorias'

class FornecedorListView(ListView):
    model = Fornecedor
    template_name = 'partials/fornecedor_list.html'
    context_object_name = 'fornecedores'
```

Template Principal dashboard.html

Neste template, inclua as listas parciais:

```
html
Copiar código
<!-- dashboard.html -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Dashboard</title>
</head>
<body>
    <h1>Produtos</h1>
    {% include 'partials/produto_list.html' %}

    <h1>Categorias</h1>
    {% include 'partials/categoria_list.html' %}

    <h1>Fornecedores</h1>
    {% include 'partials/fornecedor_list.html' %}
</body>
</html>
```

Configurando as URLs

Se precisar acessar cada ListView individualmente, configure as URLs para cada uma, mas chame apenas a DashboardView para o dashboard completo:

```
python
Copiar código
# urls.py
from django.urls import path
from .views import ProdutoListView, CategoriaListView, FornecedorListView, DashboardView

urlpatterns = [
    path('produtos/', ProdutoListView.as_view(), name='produto_list'),
    path('categorias/', CategoriaListView.as_view(), name='categoria_list'),
    path('fornecedores/', FornecedorListView.as_view(), name='fornecedor_list'),
    path('dashboard/', DashboardView.as_view(), name='dashboard'),
```

]

Resumo

1. Use TemplateView para exibir múltiplos modelos em um único template.
2. Alternativamente, crie ListView separadas para cada modelo e inclua esses templates parciais no template principal com {% include %}.

Essas abordagens permitem flexibilidade na exibição de dados de várias classes com Django e ajudam a organizar o código.