Vamos criar um exemplo simples onde vamos cadastrar e listar livros usando um **ModelForm**. A ideia é ter um projeto

A seguir, veja o passo a passo detalhado:

1. Criação do Projeto e App

1. **Crie o projeto**:

```bash
django-admin startproject meuprojeto
```

2. **Entre na pasta do projeto e crie o app**:

```bash
cd meuprojeto
python manage.py startapp minhaapp
```

3. **Adicione o app aos INSTALLED_APPS** No arquivo `meuprojeto/settings.py`, inclua `'minhaapp'` na lista de `INS`

```python
INSTALLED_APPS = [
    # apps padrões do Django
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Seu app
    'minhaapp',
]
```

2. Definindo o Modelo (models.py)

Crie um modelo simples para um livro em `minhaapp/models.py`:

```python
from django.db import models

class Livro(models.Model):
    titulo = models.CharField(max_length=100)
    autor = models.CharField(max_length=100)
    data_publicacao = models.DateField()
```

```
    def __str__(self):
        return self.titulo
```

Esse modelo possui campos para o título, o autor e a data de publicação do livro.

3. Criando o ModelForm (forms.py)

No mesmo app, crie um arquivo chamado `forms.py` para definir o formulário baseado no modelo:

Python Copy

```python
from django import forms
from .models import Livro

class LivroForm(forms.ModelForm):
    class Meta:
        model = Livro
        fields = ['titulo', 'autor', 'data_publicacao']
```

O `LivroForm` utiliza todos os campos do modelo `Livro` definidos no atributo `fields`.

4. Criando as Views (views.py)

Em `minhaapp/views.py` vamos criar duas views: uma para listar os livros cadastrados e outra para exibir e processar o

Python Copy

```python
from django.shortcuts import render, redirect
from .models import Livro
from .forms import LivroForm

def listar_livros(request):
    livros = Livro.objects.all()
    return render(request, 'minhaapp/livro_list.html', {'livros': livros})

def criar_livro(request):
    if request.method == "POST":
        form = LivroForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('listar_livros')
    else:
        form = LivroForm()
    return render(request, 'minhaapp/livro_form.html', {'form': form})
```

- `listar_livros` recupera todos os livros e os passa para o template.
- `criar_livro` exibe o formulário. Se o método for POST e o formulário for válido, o livro é salvo e o usuário é redi

5. Configurando as URLs

a) URLs do App (`minhaapp/urls.py`)

Crie um arquivo em `minhaapp/urls.py` para mapear as views:

Python Copy

```python
from django.urls import path
from .views import listar_livros, criar_livro

urlpatterns = [
    path('', listar_livros, name='listar_livros'),
    path('novo/', criar_livro, name='criar_livro'),
]
```

b) URLs do Projeto (`meuprojeto/urls.py`)

No arquivo raiz `meuprojeto/urls.py`, inclua as URLs do app:

Python Copy

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

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

Assim, as URLs para livros serão `http://localhost:8000/livros/` para a lista e `http://localhost:8000/livros/novo/`

6. Criando os Templates

Crie a estrutura de pastas para armazenar os templates do app. Por exemplo, em `minhaapp/templates/minhaapp/` crie

a) Template para a Lista de Livros (`livro_list.html`)

Html Copy

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Lista de Livros</title>
</head>
<body>
    <h1>Lista de Livros</h1>
    <ul>
        {% for livro in livros %}
            <li>{{ livro.titulo }} - {{ livro.autor }} - {{ livro.data_publicacao }}</li>
```

```
        {% empty %}
            <li>Nenhum livro cadastrado.</li>
        {% endfor %}
    </ul>
    <a href="{% url 'criar_livro' %}">Adicionar Novo Livro</a>
</body>
</html>
```

b) Template para o Formulário de Livro (`livro_form.html`)

Html Copy

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Adicionar Livro</title>
</head>
<body>
    <h1>Adicionar Livro</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Salvar Livro</button>
    </form>
    <a href="{% url 'listar_livros' %}">Voltar para a Lista</a>
</body>
</html>
```

7. Executando o Projeto

1. **Realize as migrações**:

   Bash Copy

   ```
   python manage.py makemigrations
   python manage.py migrate
   ```

2. **Crie um superusuário (opcional, para acessar a área administrativa)**:

   Bash Copy

   ```
   python manage.py createsuperuser
   ```

3. **Inicie o servidor**:

   Bash Copy

   ```
   python manage.py runserver
   ```

Agora, acesse:

- `http://localhost:8000/livros/` para visualizar a lista de livros.
- `http://localhost:8000/livros/novo/` para adicionar um novo livro.

Considerações Finais

Este exemplo demonstra uma integração simples do Django com **ModelForm**, destacando a criação do modelo, do fo

Se precisar explorar mais funcionalidades, você pode estudar como:

- Implementar formulários para atualização e exclusão (Edit/Delete).
- Utilizar mensagens para feedback dos usuários ao salvar ou atualizar dados.
- Adicionar filtros na listagem e criar páginas de detalhe para cada registro.

Espero que esse exemplo completo ajude a entender como utilizar o `ModelForm` no Django para construir aplicações

Happy coding!