Apostila: Trabalhando com Imagens no Django

1. Introdução

Neste módulo, você aprenderá a adicionar um campo de imagem ao modelo Item no Django, configurar o formulário para upload e exibir as imagens carregadas no frontend. Esta apostila inclui passos para armazenar as imagens em pastas específicas e salvar a rota da imagem no banco de dados.

Pré-requisitos

- Conhecimento básico de Django (Modelos, Views, Templates).
- Ambiente de desenvolvimento configurado com Python e Django.

2. Configurando o Ambiente para Trabalhar com Imagens

2.1 Instalando a Biblioteca Pillow

Django precisa da biblioteca Pillow para trabalhar com imagens. Para instalar, execute:

bash
Copiar código
pip install Pillow

3. Modificando o Modelo Item para Incluir o Campo de Imagem

Vamos modificar o modelo Item para incluir um campo de imagem usando o ImageField.

Exemplo de Código

No arquivo models.py dentro do aplicativo app, atualize o modelo Item:

python
Copiar código

```
from django.db import models

class Item(models.Model):
    nome = models.CharField(max_length=100)
    descricao = models.TextField()
    preco = models.DecimalField(max_digits=10, decimal_places=2)
    criado_em = models.DateTimeField(auto_now_add=True)
    imagem = models.ImageField(upload_to='itens_imagens/', null=True, blank=True)

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

Explicação dos Campos:

- imagem: Campo ImageField com upload_to='itens_imagens/', especificando a pasta onde as imagens serão armazenadas.

Configurando o Diretório de Mídia

No settings.py do projeto, adicione a configuração para o diretório de mídia:

python
Copiar código
```
import os

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

Essas configurações definem onde as imagens e outros arquivos de mídia serão salvos.

Configurando as URLs para Arquivos de Mídia

No urls.py principal do projeto, adicione as seguintes linhas para habilitar o acesso aos arquivos de mídia durante o desenvolvimento:

python
Copiar código
```
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    # Suas outras rotas...
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

> Nota: Essas configurações são recomendadas apenas para o ambiente de desenvolvimento. Em produção, é necessário configurar o servidor para servir arquivos de mídia.

4. Criando o Formulário

Agora, vamos criar um formulário para permitir o upload de imagens no Django.

Exemplo de Formulário com ModelForm

No arquivo forms.py dentro do aplicativo app, crie o formulário ItemForm:

python
Copiar código
```
from django import forms
from .models import Item

class ItemForm(forms.ModelForm):
    class Meta:
        model = Item
        fields = ['nome', 'descricao', 'preco', 'imagem']
```

5. Criando a View para o Formulário de Upload

No arquivo views.py dentro do aplicativo app, crie uma view para processar o formulário.

python
Copiar código

```
from django.shortcuts import render, redirect
from .forms import ItemForm

def upload_item(request):
    if request.method == 'POST':
        form = ItemForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('itens_listagem')  # Redireciona após o upload bem-sucedido
    else:
        form = ItemForm()
    return render(request, 'upload_item.html', {'form': form})
```

Nesta view:

- request.FILES permite o upload de arquivos (imagens).
- form.save() salva o objeto Item com a imagem carregada.

6. Criando o Template com variavel.as_p

Vamos criar um template HTML para exibir o formulário de upload. Usaremos as_p para renderizar o formulário.

Exemplo de Código HTML (Template)

Crie o arquivo upload_item.html em templates/app:

html
Copiar código

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Upload de Item</title>
</head>
<body>
    <h1>Upload de Item</h1>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Salvar</button>
    </form>
</body>
</html>
```

Explicação:

- {% csrf_token %}: Inclui o token CSRF para segurança.
- {{ form.as_p }}: Renderiza o formulário em parágrafos <p>.

7. Exibindo os Itens e as Imagens Carregadas

Vamos criar uma view e um template para exibir a lista de itens com as imagens.

View para Listagem de Itens

No arquivo views.py, adicione uma view para listar os itens:

python
Copiar código

```
from .models import Item

def itens_listagem(request):
    itens = Item.objects.all()
    return render(request, 'itens_listagem.html', {'itens': itens})
```

Template para Listagem de Itens

Crie o arquivo itens_listagem.html em templates/app:

html
Copiar código

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Listagem de Itens</title>
</head>
<body>
    <h1>Itens</h1>
    <ul>
        {% for item in itens %}
            <li>
                <strong>{{ item.nome }}</strong><br>
                <p>{{ item.descricao }}</p>
                <p>Preço: R$ {{ item.preco }}</p>
                {% if item.imagem %}
                    <img src="{{ item.imagem.url }}" alt="{{ item.nome }}" width="150" height="150">
                {% else %}
                    <p>Imagem não disponível</p>
                {% endif %}
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

Explicação:

- {% for item in itens %}: Itera sobre todos os itens.
- {{ item.imagem.url }}: Exibe a imagem do item.

8. Testando o Sistema

1. Acesse a URL configurada para o upload e faça o upload de imagens para os itens.

2. Navegue até a página de listagem para confirmar se os itens, incluindo as imagens, são exibidos corretamente.

9. Conclusão

Você agora configurou com sucesso o campo ImageField no modelo Item, criou o formulário ItemForm para o upload de imagens e configurou templates para exibir os itens com as imagens. Essa estrutura é fundamental para lidar com arquivos de mídia em projetos Django.