Blogging Platform using Django

Last Updated : 24 Sep, 2024

foundation to build upon. In this article, we will explore the process of building a blogging platform using Django, covering key concepts and steps along the way.

Blogging Platform using Django

A blogging platform using Django is a web application built with the Django framework that allows users to create, publish, and manage blog posts on a website, providing a user-friendly content management system for bloggers.

Setting Up the Project Folder

```
django-admin startproject <project_name>
cd <project_name>
python manage.py startapp gallery
```

model.py: Here we have created a Product table with name, description, image, created_at, and updated_at fields in the table.

Python

```python
from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    image = models.ImageField(upload_to='products/')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name

    def edit(self, name, description, image):
        self.name = name
        self.description = description
        self.image = image
        self.save()


    def short_description(self):
        # Split the description into words
        words = self.description.split()
        if len(words) > 50:
            # Join the first 50 words and add "..." at the end
            return ' '.join(words[:30]) + '...'
        else:
            # If the description is already less than 50 words, return it as is
            return self.description
```

admin.py: Here we are registering our table in the admin.

Python

```
from django.contrib import admin
from .models import Product

# Register your models here.
admin.site.register(Product)
```

views.py: Here's a brief explanation of each part:

- product_list(request): This view retrieves a list of all products from the database using the Product model and renders a template named 'index.html', passing the list of products to the template.
- product_detail(request, pk): This view retrieves a specific product by its primary key (pk) from the database and renders a template named 'index2.html', passing the product data to the template.

- edit_product(request, pk): This view handles editing a product. It retrieves the product to edit, processes a submitted form to update the product's information, and redirects to the product list if the form is valid. Otherwise, it displays the form for editing.
- delete_product(request, pk): Similar to the edit view, this view handles product deletion. If a POST request is made, it deletes the product and redirects to the product list. Otherwise, it displays a confirmation page for deleting the product.

Python

```python
from django.shortcuts import render
from .models import Product
from django.http import HttpResponse
from django.shortcuts import render, redirect, get_object_or_404
from .models import Product
from .forms import ProductForm


def product_list(request):
    products = Product.objects.all()
    return render(request, 'myapp/index.html', {'products': products})

def product_detail(request, pk):
    product = Product.objects.get(pk=pk)
    return render(request, 'myapp/index2.html', {'product': product})


def edit_product(request, pk):
    product = get_object_or_404(Product, pk=pk)
    if request.method == 'POST':
        form = ProductForm(request.POST, instance=product)
        if form.is_valid():
            form.save()
            return redirect('product_list')
    else:
        form = ProductForm(instance=product)
    return render(request, 'myapp/edit.html', {'form': form})

def delete_product(request, pk):
    product = get_object_or_404(Product, pk=pk)
    if request.method == 'POST':
        product.delete()
        return redirect('product_list')
    return render(request, 'myapp/delete.html', {'product': product})


def home(request):
    return HttpResponse('Hello, World!')
```

form.py: Here we created a form to edit the blog when a user requests.

Python

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

class ProductForm(forms.ModelForm):
    class Meta:
        model = Product
        fields = ['name', 'description', 'image']
```

Image Gallery GUI

In the product list template, it loops through the products queryset and displays each product's name, description, and an image. The blog detail template shows detailed information about each blog and provides a link to go back to the blog home page.

edit.html: HTML file to edit the Blog.

HTML

```html
<h2>Edit Blog Post</h2>
```

```
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save Changes</button>
</form>
```

delete.html: HTML file to delete the Blog.

HTML

```
<h2>Delete Blog Post</h2>
<p>Are you sure you want to delete "{{ product.name }}"?</p>
<form method="post">
    {% csrf_token %}
    <button type="submit">Yes, Delete</button>
</form>
```

index.html: HTML file to show all the list of the Blog.

Python

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blog Website</title>
    <!-- Add Bootstrap CSS link here -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h1>Today's New Blogs</h1>
        <div class="row">
            {% for product in products %}
                <div class="col-md-4 mb-4">
                    <div class="card">
```

```
                        <a href="{% url 'product_detail' product.pk %}">
                            <img src="{{ product.image.url }}" alt="{{ product.created_name }}" class="card-img-top">
                        </a>
                        <div class="card-body">
                            <h5 class="card-title">{{ product.name }}</h5>
                            <p class="card-text">{{ product.short_description }}</p>
                            <a href="{% url 'edit_product' product.pk %}">Edit</a>
                            <a href="{% url 'delete_product' product.pk %}">Delete</a> <br>
                            <small class="card-text">Created at: {{ product.created_at }}</small> <br>
                            <small class="card-text">Updated at: {{ product.updated_at }}</small>
                        </div>
                    </div>
                </div>
            {% endfor %}
        </div>
    </div>

    <!-- Add Bootstrap JS and jQuery scripts here (if needed) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>
```

index2.html: HTML file to show the full details of the Blog.

Python

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{ product.name }} - Blog Detail</title>
</head>
<body>
    <h1>{{ product.name }} - Blog </h1>

    <div>
        <img src="{{ product.image.url }}" alt="{{ product.name }}" width="200">
    </div>
    <h2>{{ product.name }}</h2>
    <p>{{ product.description }}</p>
    <a href="{% url 'edit_product' product.pk %}">Edit</a>
    <a href="{% url 'delete_product' product.pk %}">Delete</a> <br>

    <a href="{% url 'product_list' %}">Back to Blogs Home Page</a>
</body>
</html>
```

urls.py: Define the URL patterns in the urls.py file of the catalog app to map views to URLs.

Python

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

urlpatterns = [
    path('/home', views.home, name='home'),
    path('', views.product_list, name='product_list'),
    path('<int:pk>/', views.product_detail, name='product_detail'),
    path('<int:pk>/edit/', views.edit_product, name='edit_product'),
    path('<int:pk>/delete/', views.delete_product, name='delete_product'),
]
```

urls.py: Add the necessary URL patterns in your project's urls.py.

Python

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

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

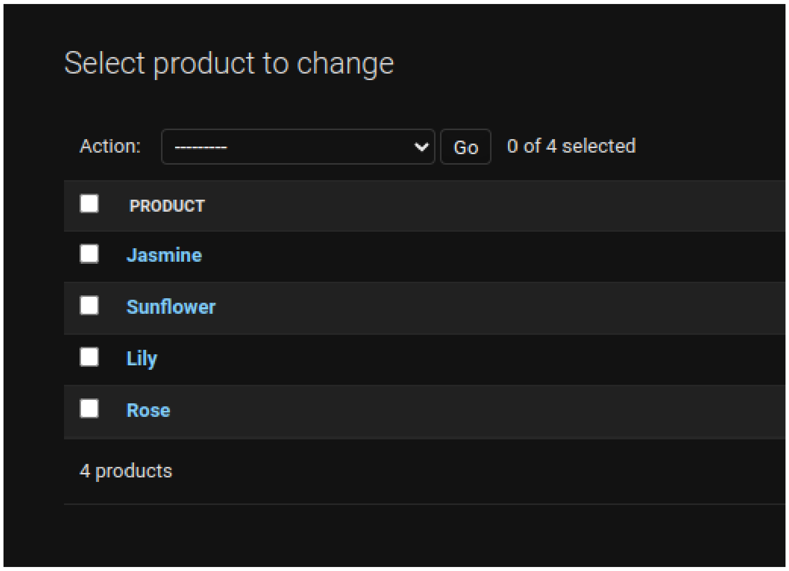
Building a blogging platform is a great project for learning Django. If you're looking to take your Django skills to the next level, consider the **Django Web Development Course** to gain more insight into advanced development techniques.

Deploy the Project

Create a super user to add data into the data base by the following command:

```
python manage.py createsuperuser
```

Now, Go to the http://127.0.0.1:8000/admin/ and add the Images, name and its description.



Select product to change

Action: --------- Go 0 of 4 selected

□	PRODUCT
□	Jasmine
□	Sunflower
□	Lily
□	Rose

4 products

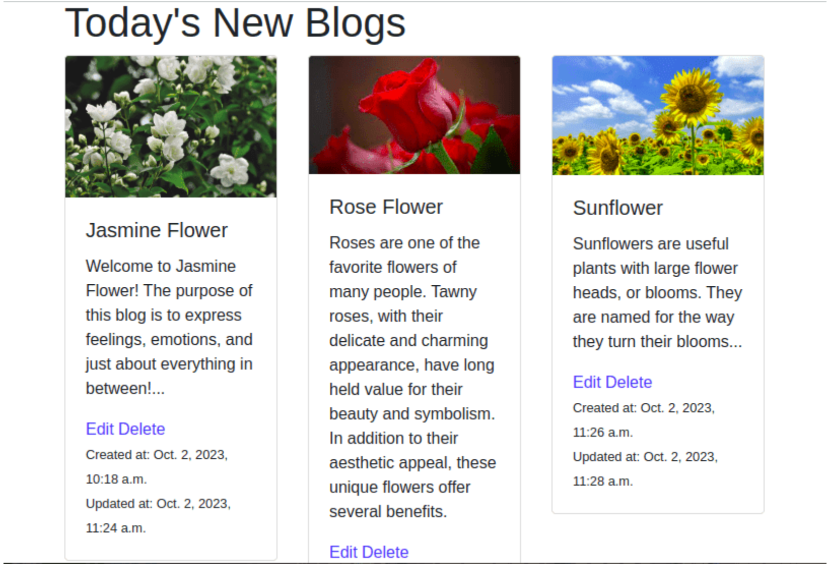
Migrate the data into the database.

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

Deploy the project

```
python manage.py runserver
```

Output:



Today's New Blogs

Jasmine Flower

Welcome to Jasmine Flower! The purpose of this blog is to express feelings, emotions, and just about everything in between!...

Edit Delete

Created at: Oct. 2, 2023, 10:18 a.m.

Updated at: Oct. 2, 2023, 11:24 a.m.

Rose Flower

Roses are one of the favorite flowers of many people. Tawny roses, with their delicate and charming appearance, have long held value for their beauty and symbolism. In addition to their aesthetic appeal, these unique flowers offer several benefits.

Edit Delete

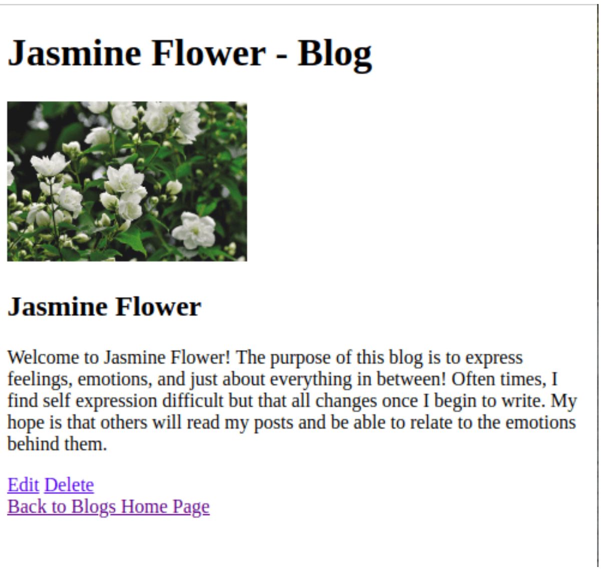
Sunflower

Sunflowers are useful plants with large flower heads, or blooms. They are named for the way they turn their blooms...

Edit Delete

Created at: Oct. 2, 2023, 11:26 a.m.

Updated at: Oct. 2, 2023, 11:28 a.m.

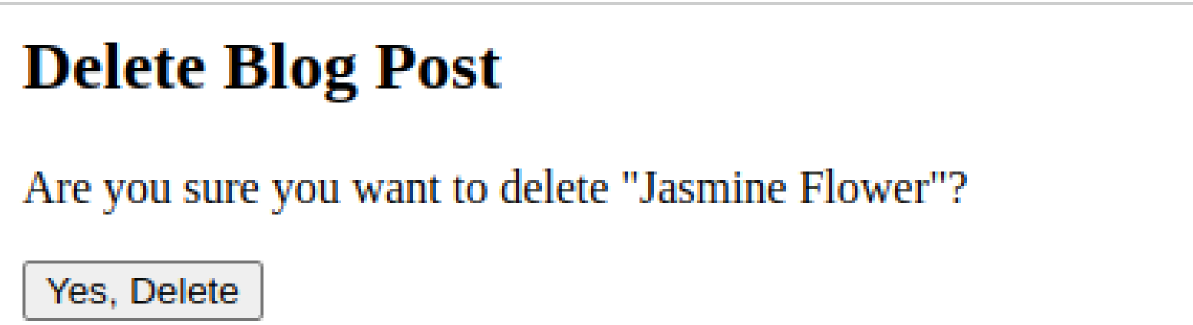


Jasmine Flower - Blog

Jasmine Flower

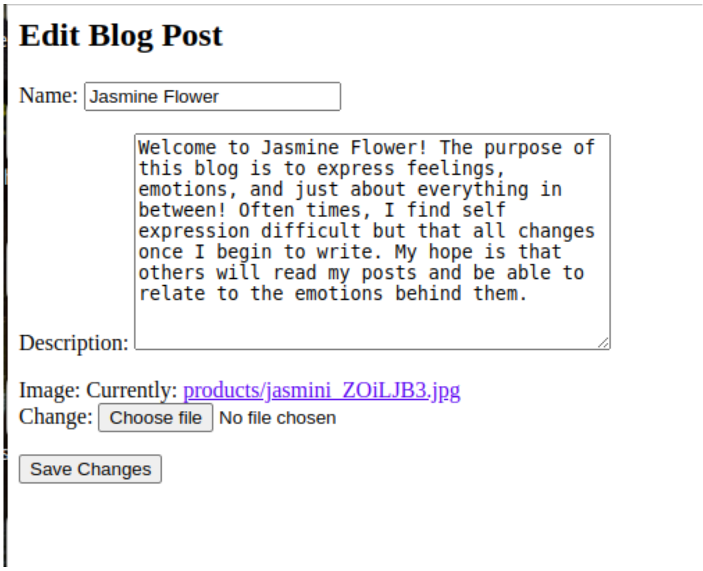
Welcome to Jasmine Flower! The purpose of this blog is to express feelings, emotions, and just about everything in between! Often times, I find self expression difficult but that all changes once I begin to write. My hope is that others will read my posts and be able to relate to the emotions behind them.

Edit Delete
Back to Blogs Home Page



Delete Blog Post

Are you sure you want to delete "Jasmine Flower"?



Edit Blog Post

Name: Jasmine Flower

Description: Welcome to Jasmine Flower! The purpose of this blog is to express feelings, emotions, and just about everything in between! Often times, I find self expression difficult but that all changes once I begin to write. My hope is that others will read my posts and be able to relate to the emotions behind them.

Image: Currently: products/jasmini_ZOiLJB3.jpg
Change: Choose file No file chosen

A amit... 4

Previous Article

E-commerce Product Catalog using Django

Next Article

Similar Reads

Blogging Platform using Next JS

In this project, we will explore the process of building The Blogging Platform with Next.js. Blogging Platform is a web application that allows users to create...

5 min read

Blogging Platform using MERN Stack

The blogging platform is developed using the MERN (MongoDB, ExpressJS, ReactJS, NodeJS) stack, allowing users to create, read, update, and delete blog...

5 min read

RESTful Blogging API with Node and Express.js

Blogs Websites have become very popular nowadays for sharing your thoughts among the users over internet. In this article, you will be guided through creatin...

6 min read

Property Listing Platform using Node and ExpressJS

This tutorial can help you build a foundational understanding of the backend development. NodeJS and ExpressJS are used to build the backend infrastructur...

6 min read

Project Idea | Build an AR based music learning platform using the MERN...

In this article, we will be building an AR-based web app called GuitAR. It teaches users to learn guitar by projecting the strings that are to be plucked on the guita...

15+ min read

Ticket Raising Platform using MERN Stack

The Ticket Raising Platform project is a web application that enables users to create, manage, and track tickets for various issues or tasks. Built using the MER...

8 min read

Real Estate Listings Platform using NextJS

In this article, we will explore the process of building a real estate listing platform using Next.js. real estate listings platform is a web application that aims to...

6 min read

Music Streaming Platform using MERN Stack

In this tutorial, we'll walk through the process of creating a music streaming platform using the MERN stack. This project will allow users to browse playlists...

7 min read

Social Networking Platform using Next.js

The Social Networking Platform built with NextJS is a web application that provides users the functionality to add a post, like a post, and be able to comme...

8 min read

Product Review Platform using MEAN Stack

In today's digital age, online reviews play an important role in shaping consumer decisions. Whether it's choosing a restaurant, purchasing a gadget, or booking a...

15+ min read

Article Tags : Django Geeks Premier League Project Geeks Premier League 2023

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305

Company
About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

Languages
Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA
Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML
Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies
HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial
Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects