Django – Blogs App (Part – 2)

Example: Adding forms to blogs app.

Create a database model for blogs app

blog/models.py

```
from django.db import models
from django.utils import timezone
from django.urls import reverse

class Post(models.Model):
    author = models.ForeignKey('auth.User', on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(
            default=timezone.now)
    published_date = models.DateTimeField(
            blank=True, null=True)

    def publish(self):
        self.published_date = timezone.now()
        self.save()

    def __str__(self):
        return self.title

    # when form is submitted it takes you to mentioned url
    def get_absolute_url(self):
        return reverse('post_detail', args=[str(self.id)])
```

Views/URLs (Class-Based)

blog/urls.py

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

urlpatterns = [
    path('post/new/', views.BlogCreateView.as_view(), name='post_new'),
    path('post/<int:pk>/edit/', views.BlogUpdateView.as_view(), name='post_edit'),
    path('post/<int:pk>/delete/', views.BlogDeleteView.as_view(), name='post_delete'),

# (Continued from previous module)
    path('', views.BlogListView.as_view(), name='viewblogs'),
    path('post/<int:pk>/', views.BlogDetailView.as_view(), name='post_detail'),]
```

blog/views.py

```
from django.views.generic import ListView, DetailView, CreateView,UpdateView,DeleteView
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
from django.urls import reverse_lazy

from .models import Post
from .forms import PostForm, EditPostForm

# To add new object
class BlogCreateView(CreateView):
    model = Post
    template_name = 'post_new.html'
    form_class = PostForm

    def form_valid(self, form):
        form.instance.publish()
        return super().form_valid(form)

# To update object
class BlogUpdateView(UpdateView):
    model = Post
    form_class = PostForm
    template_name = 'post_edit.html'

# Delete a Post(always works with post method)
class BlogDeleteView(DeleteView):
    model = Post
    # template_name = None
    success_url = reverse_lazy('viewblogs')

# List all the Posts (Continued from previous module)

class BlogListView(ListView):
    model = Post
    context_object_name = "posts"
    template_name = "listblogs.html"

# View a specific post (Continued from previous module)
class BlogDetailView(DetailView):
    model = Post
    template_name = 'post_detail.html'
```

Views/URLs (Function-Based)

blog/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('post/new', views. blogCreateView, name='post_new'),
    path('post/<int:pk>/edit/', views. blogUpdateView, name='post_edit'),
    path('post/<int:pk>/delete/', views. blogDeleteView, name='post_delete'),

# (Continued from previous module)
    path('', views.blogListView, name='viewblogs'),
    path('post/<int:pk>/', views.blogDetailView, name='post_detail'),
]
```

blog/forms.py

```
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    title = forms.CharField(widget=forms.TextInput(attrs={
        'class' : 'form-control'
    }))
    text  = forms.CharField(widget=forms.Textarea(attrs={
        'class' : 'form-control'
    }))
    class Meta:
        model = Post
        fields = ('author', 'title', 'text',)

class EditPostForm(forms.ModelForm):
    title = forms.CharField(widget=forms.TextInput(attrs={
        'class' : 'form-control'
    }))
    text  = forms.CharField(widget=forms.Textarea(attrs={
        'class' : 'form-control'
    }))
    class Meta:
        model = Post
        fields = ('title', 'text',)
```

blog/views.py

```
from django.views.generic import ListView, DetailView, CreateView,UpdateView,DeleteView
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
from django.urls import reverse_lazy

from .models import Post
from .forms import PostForm, EditPostForm
```

```
# To add new object
def blogCreateView (request):

    if request.method == "POST":
        form = PostForm(request.POST)
        if form.is_valid():
            post = form.save(commit=False)
            post.publish()
            post.save()
            return redirect('post_detail', pk=post.pk)
    else:
        form = PostForm()
    return render(request, 'post_new.html', {'form': form})

# To update object
def blogUpdateView (request, pk):
    post = get_object_or_404(Post, pk=pk)
    if request.method == "POST":
        form = EditPostForm(request.POST, instance=post)
        if form.is_valid():
            post = form.save(commit=False)
            post.save()
            return redirect('post_detail', pk=post.pk)
    else:
        form = EditPostForm(instance=post)
    return render(request, 'post_edit.html', {'form': form})

# To delete object
def blogDeleteView (request, pk):
    post= get_object_or_404(Post, pk=pk)
    post.delete()
    return redirect('viewblogs')

# List all the Posts  (Continued from previous module)
def blogListView(request):
    posts = Post.objects.filter(published_date__lte=timezone.now()).order_by('published_date')
    return render(request, 'listblogs.html', {'posts': posts})

# View a specific post  (Continued from previous module)
def blogDetailView(request, pk):
    post = get_object_or_404(Post, pk=pk)
    return render(request, 'post_detail.html', {'post': post})
```

Templates

blog/templates/base.html

```
{% load static %}

<html>
<head>
    <title>My Shopping Blog</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
    <link href='//fonts.googleapis.com/css?family=Lobster&subset=latin,latin-ext' rel='stylesheet'
type='text/css'>
    <link rel="stylesheet" href="{% static 'css/base.css' %}">
</head>
<body>
<div class="content container">
    <div class="page-header">
        <h1>My Shopping Blog</h1>
        <a href="/blog">Home</a> | <a href="{% url 'post_new' %}"> Add new post</a>
    </div>
    {% block content %}
    {% endblock content %}
</div>
</body>
</html>
```

blog/templates/listblogs.html

```
{% extends 'base.html' %}
{% block content %}
  <div class="content container">
            <div class="row">
                <div class="col-md-8">
                {% for post in posts %}
                    <div class="post">
                        <h3><a href="{% url 'post_detail' post.pk %}">{{ post.title }}</a></h3>
                        <!--<h1><a href="post/{{post.pk}}">{{ post.title }}</a></h1>-->
                        <div class="date">
                            {{ post.published_date }}
                        </div>
                    </div>
                {% endfor %}
                </div>
            </div>
        </div>
{% endblock content %}
```

blog/templates/post_detail.html

```
{% extends 'base.html' %}
{% block content %}
<div class="content container">
    <div class="row">
        <div class="col-md-8">
            <div class="post">
                <div class="date">
                    {{ post.published_date }}
                </div>
                <h2>{{ post.title }}</h2>
                <p>{{ post.text|linebreaksbr }}</p>
            </div>
            <form action="{% url 'post_delete' post.pk %}" method="post">{% csrf_token %}
                <a href="{% url 'post_edit' post.pk %}" class="btn btn-outline-primary btn-sm">Edit</a> |
<input type="submit" value="Delete" class="btn btn-outline-primary btn-sm">
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

blog/templates/post_new.html

```
{% extends 'base.html' %}
{% block content %}
<div class="content container">
    <div class="col-md-8">
        <h3>New post</h3>
        <form action="" method="post">{% csrf_token %}
            {{ form.as_table }}
            <input type="submit" value="Save"/>
        </form>
    </div>
</div>
{% endblock %}
```

blog/templates/post_edit.html

```
{% extends 'base.html' %}
{% block content %}
<div class="content container">
    <div class="col-md-8">
        <h3>Edit post</h3>
        <form action="" method="post">{% csrf_token %}
            {{ form.as_table }}
            <input type="submit" value="Update"/>
        </form>
    </div>
</div>
{% endblock %}
```

Tests

blog/tests.py

```
from django.contrib.auth import get_user_model
from django.test import Client, TestCase
from django.urls import reverse
from .models import Post

class BlogTests(TestCase):

    def setUp(self):
        self.user = get_user_model().objects.create_user(
            username='testuser',
            email='test@email.com',
            password='secret'
        )

        self.post = Post.objects.create(
            title='A good title',
            text='Nice body content',
            author=self.user,
        )

# Test - 1 (Continued from previous module)
    def test_string_representation(self):
        post = Post(title='A sample title')
        self.assertEqual(str(post), post.title)

# Test - 2 (Continued from previous module)
    def test_post_content(self):
        self.assertEqual(str(self.post.title), 'A good title')
        self.assertEqual(str(self.post.author), 'testuser')
        self.assertEqual(str(self.post.text), 'Nice body content')

# Test - 3 (Continued from previous module)
    def test_post_list_view(self):
        response = self.client.get(reverse('viewblogs'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'listblogs.html')
        self.assertContains(response, 'A good title')

# Test - 4 (Continued from previous module)
    def test_post_detail_view(self):
        no_response = self.client.get('/blog/post/100000/')
        self.assertEqual(no_response.status_code, 404)

        response = self.client.get('/blog/post/1/')
        self.assertEqual(response.status_code, 200)

        self.assertContains(response, 'Nice body content')
        self.assertTemplateUsed(response, 'post_detail.html')
```

```
# Test - 5
    def test_post_create_view(self):
        response = self.client.post(reverse('post_new'), {
            'title': 'New title',
            'body': 'New text',
            'author': self.user,
        })
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, 'New title')

# Test - 6
    def test_post_update_view(self):
            response = self.client.post(reverse('post_edit', args='1'), {
                'title': 'Updated title',
                'body': 'Updated text',
            })
            self.assertEqual(response.status_code, 200)
            self.assertContains(response, 'Updated title')

# Test - 7
    def test_post_delete_view(self):
         response = self.client.post(reverse('post_delete', args='1'))
         print(response)
         self.assertEqual(response.status_code, 302)

# Test - 8
    def test_get_absolute_url(self):
            self.assertEqual(self.post.get_absolute_url(), '/blog/post/1/')
```