Django – Blogs App (Part – 1)

Example: Adding blogs app to our site.

Create a database model for blogs app

blog/models.py

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

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
```

Views/URLs (Class-Based)

blog/urls.py

```
from . import views
urlpatterns = [
    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
from . models import Post

# List all the Posts
class BlogListView(ListView):
    model = Post
    context_object_name = "posts"
    template_name = "listblogs.html"
```

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

Views/URLs (Function-Based)

blog/urls.py

```
from . import views
urlpatterns = [
    path('', views.blogListView, name='viewblogs'),
    path('post/<int:pk>/', views.blogDetailView, name='post_detail'),
]
```

blog/views.py

```
from django.shortcuts import render, get_object_or_404
from django.utils import timezone
from . models import Post

# List all the Posts
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
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="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
        <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap-theme.min.css">
        <link href='//fonts.googleapis.com/css?family=Lobster&subset=latin,latin-ext' rel='stylesheet'
type='text/css'>
        <link rel="stylesheet" href="{% static 'css/blog.css' %}">
    </head>
    <body>
        <div class="page-header">
            <h1><a href="/blog">My Shopping Blog</a></h1>
        </div>
      {% block content %}
      {% endblock content %}
  </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">
                        <h1><a href="{% url 'post_detail' post.pk %}">{{ post.title }}</a></h1>
                        <!--<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">
        <h6><a href="/blog">Home</a></h6>
            <div class="row">
                <div class="post">
                    {% if post.published_date %}
                        <div class="date">
```

```
                            {{ post.published_date }}
                        </div>
                    {% endif %}
                    <h1>{{ post.title }}</h1>
                    <p>{{ post.text|linebreaksbr }}</p>
                </div>
            </div>
    </div>
{% endblock %}
```

Static Files

Just as we did with our templates folder we need to update settings.py to tell Django where to look for static files. We can update settings.py with a one-line change for STATICFILES_DIRS. Add it at the bottom of the file below the entry for STATIC_URL.

mysite/settings.py

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
```

Now create a folders /static/css and add a new base.css file in it.

static/css/base.css

```
body {
  font-family: 'Source Sans Pro', sans-serif;
  font-size: 18px;
}

header {
  border-bottom: 1px solid #999;

margin-bottom: 2rem;
  display: flex;
}
.
.
.
```

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
    def test_string_representation(self):
        post = Post(title='A sample title')
        self.assertEqual(str(post), post.title)
# Test - 2
    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
    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
    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')
```