Part 10 – Create, Update and Delete Posts

**urls.py**

from django.urls import path

from . import views

from .views import (

PostListView, PostDetailView, PostCreateView,

PostUpdateView, PostDeleteView

)

urlpatterns = [

path('', PostListView.as\_view(), name='blog-home'),

path('post/<int:pk>/', PostDetailView.as\_view(), name='post-detail'),

path('post/new/', PostCreateView.as\_view(), name='post-create'),

path('post/<int:pk>/update/', PostUpdateView.as\_view(),

name='post-update'),

path('post/<int:pk>/delete/', PostDeleteView.as\_view(),

name='post-delete'),

path('about/', views.about, name='blog-about'),

]

**views.py**

from django.shortcuts import render

from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin

from django.views.generic import (

ListView, DetailView, CreateView,

UpdateView, DeleteView

)

from .models import Post

class PostListView(ListView):

model = Post

template\_name = 'blog/home.html' # <app>/<model>\_<viewtype>.html

context\_object\_name = 'posts' # default: object\_list

ordering = ['-date\_posted']

class PostDetailView(DetailView):

model = Post

class PostCreateView(LoginRequiredMixin, CreateView):

model = Post

fields = ['title', 'content']

# You can get the current user with self.request.user in the view.

# form.instance.author 🡪 same as PostUpdateView Below

# e.g. PostForm(request.POST or None, instance=blahBlah)

def form\_valid(self, form):

form.instance.author = self.request.user

return super().form\_valid(form)

class PostUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):

model = Post

fields = ['title', 'content']

def form\_valid(self, form):

form.instance.author = self.request.user

return super().form\_valid(form)

# Model.objects.all()

self.get\_object() is the one currentobject of Model.objects.all

def test\_func(self):

post = self.get\_object()

if self.request.user == post.author:

return True

return False

class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):

model = Post

success\_url = '/'

def test\_func(self):

post = self.get\_object()

if self.request.user == post.author:

return True

return False

**home.html**

{% extends "blog/base.html" %}

{% block content %}

{% for post in posts %}

<article class="media content-section">

<img class="rounded-circle article-img"

src="{{ post.author.profile.image.url }}">

<div class="media-body">

<div class="article-metadata">

<a class="mr-2" href="#">{{ post.author }}</a>

<small class="text-muted">

{{ post.date\_posted|date:"F d, Y" }}</small>

</div>

<h2><a class="article-title"

href="{% url 'post-detail' post.id %}">

{{ post.title }}</a>

</h2>

<p class="article-content">{{ post.content }}</p>

</div>

</article>

{% endfor %}

{% endblock content %}

**post\_detail.html**

{% block content %}

<article class="media content-section">

<img class="rounded-circle article-img"

src="{{ object.author.profile.image.url }}">

<div class="media-body">

<div class="article-metadata">

<a class="mr-2" href="#">{{ object.author }}</a>

<small class="text-muted">

{{ object.date\_posted|date:"F d, Y" }}</small>

{% if object.author == user %}

<div>

<a class="btn btn-secondary btn-sm mt-1 mb-1"

href="{% url 'post-update' object.id %}">Update</a>

<a class="btn btn-danger btn-sm mt-1 mb-1"

href="{% url 'post-delete' object.id %}">Delete</a>

</div>

{% endif %}

</div>

<h2 class="article-title">{{ object.title }}</h2>

<p class="article-content">{{ object.content }}</p>

</div>

</article>

**post\_form.html**

{% extends "blog/base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4">Blog Post</legend>

{{ form|crispy }}

</fieldset>

<div class="form-group">

<button class="btn btn-outline-info"

type="submit">Post</button>

</div>

</form>

</div>

{% endblock content %}

**post\_delete.html**

{% extends "blog/base.html" %}

{% block content %}

<div class="content-section">

<form method="POST">

{% csrf\_token %}

<fieldset class="form-group">

<legend class="border-bottom mb-4">Delete Post</legend>

<h2>Are you sure you want to delete the post "

{{ object.title }}"</h2>

</fieldset>

<div class="form-group">

<button class="btn btn-outline-danger" type="submit">

Yes, Delete</button>

<a class="btn btn-outline-secondary"

href="{% url 'post-detail' object.id %}">Cancel</a>

</div>

</form>

</div>

{% endblock content %}

\_\_\_\_\_\_\_\_\_\_\_\_\_

In object-oriented programming languages, a **mixin**(or mix-in) **is** a class that contains methods for use by other classes without having to be the parent class of those other classes.