Web Development with Django framework

Instructor :Kamalpreet Kaur Bagral

Set up a Django environment.

- **Prerequisites**

✓Python installed (3.6, 3.7, 3.8, or 3.9)
✓pip (Python package installer)
✓Virtual environment package (optional but recommended)

Step-by-Step Guide

- **Install Python**:

✓Make sure Python is installed. You can download it from python.org.

✓Verify the installation by running:

❑python –version

✓**Install pip**:

✓pip should come installed with Python. Verify it by running:

❑pip --version

Set Up a Virtual Environment (Optional but Recommended):

- Install ‘ **virtualenv’**

➢**pip install virtualenv**

- Create a virtual environment:

➢**python -m venv myenv**

- Activate the virtual environment

➢On Windows

➢**myenv\Scripts\activate**

- On macOS/Linux

➢**source myenv/bin/activate**

Install Django:

- Install Django using pip:

➢pip install Django

- Verify the installation

➢python -m django –version

Create a Django Project:

- **Create a new Django project**

❑django-admin startproject myproject

- **Navigate into the project directory**

❑cd myproject

- **Run the Development Server**:

❑python manage.py runserver

Basic Project Structure

- myproject/
  manage.py
  myproject/
    __init__.py
    settings.py
    urls.py
    wsgi.py

Creating an App

- **Create an app within your project**:

✓python manage.py startapp myapp

- **Add the App to the Project**:

✓ INSTALLED_APPS = [

    ...

    'myapp',

]

Problem Statement

Develop a Simple Blog Application with Views, URL Mapping, and HTTP Handling

Objective:

Build a simple blog application that allows users to view a list of blog posts, read individual posts, and submit new posts. Implement views, URL mappings, view logic, and handle HTTP requests and responses

Requirements:

1. **Set Up Basic Structure:**
    - Create a basic web application structure with a home page, a list of blog posts, and a form to submit new posts.
2. **View Blog Posts:**
    - Create a view that lists all blog posts with their titles and excerpts.
    - Each post should have a link to a detailed view.
3. **View Individual Post:**
    - Create a view that displays the full content of an individual blog post when a user clicks on its title.
    - Include navigation to return to the list of blog posts.

4. **Create New Post:**

- Create a form for submitting a new blog post, including fields for title and content.
- Handle form submission via HTTP POST request and display a success message or errors.

5. **Handle URL Mapping:**

- Map URLs to the corresponding views for listing posts, viewing individual posts, and submitting new posts.

6. **HTTP Requests and Responses:**

- Handle HTTP GET requests for displaying blog posts and individual post details.
- Handle HTTP POST requests for submitting new posts.

1. **Homepage (`/`):**
   - Display a welcome message and a link to the list of blog posts.
2. **List Posts (`/posts`):**
   - Display a list of blog posts with titles and excerpts.
   - Each title should link to the full post view.
3. **View Post (`/posts/<id>`):**
   - Display the full content of the selected post.
   - Include a back button to return to the list of posts.
4. **New Post (`/posts/new`):**
   - Display a form for submitting a new blog post.
   - Process form submission and add the new post to the list.
   - Show success or error messages based on validation.

URL Mapping and Views

urls.py

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

urlpatterns = [
    path('', views.home, name='home'),
    path('posts/', views.list_posts, name='list_posts'),
    path('posts/<int:id>/', views.view_post, name='view_post'),
    path('posts/new/', views.new_post, name='new_post'),
]
```

views.py

```
from django.shortcuts import render, redirect
from .models import Post
from .forms import PostForm

def home(request):
    return render(request, 'home.html')

def list_posts(request):
    posts = Post.objects.all()
    return render(request, 'list_posts.html', {'posts': posts})

def view_post(request, id):
    post = Post.objects.get(id=id)
    return render(request, 'view_post.html', {'post': post})

def new_post(request):
    if request.method == 'POST':
        form = PostForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('list_posts')
```

Templates—-.
home.html

- <!DOCTYPE html>
- <html lang="en">
- <head>
- <meta charset="UTF-8">
- <title>Home</title>
- </head>
- <body>
- <h1>Welcome to the Blog</h1>
- View Blog Posts
- </body>
- </html>
-

list_posts.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Blog Posts</title>
</head>
<body>
    <h1>Blog Posts</h1>
    <ul>
        {% for post in posts %}
            <li>
                <a href="{% url 'view_post' post.id %}">{{ post.title }}</a>
                <p>{{ post.excerpt }}</p>
            </li>
        {% endfor %}
    </ul>
    <a href="{% url 'new_post' %}">New Post</a>
</body>
```

view_post.html

- <!DOCTYPE html>
- <html lang="en">
- <head>
- <meta charset="UTF-8">
- <title>{{ post.title }}</title>
- </head>
- <body>
- <h1>{{ post.title }}</h1>
- {{ post.content }}
- Back to Posts
- </body>
- </html>
-

new_post.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>New Post</title>
</head>
<body>
    <h1>Create a New Post</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Submit</button>
    </form>
    <a href="{% url 'list_posts' %}">Back to Posts</a>
</body>
</html>
```