TRAINING DAY4 REPORT

27 JUNE 2025

What is a View in Django?

A **view** in Django is a function or class that **responds to a web request** with a **web response**. It is the **core of Django's MVT (Model-View-Template)** architecture.

- The view is responsible for handling the **business logic** of your web page.
- It can retrieve data from the database, perform calculations, and return results to the user, often by rendering an HTML template.

Role of Views in Django Architecture

- Django follows the **MVT pattern**:
- **Model** → Handles data
- **View** → Processes data and logic
- **Template** → Displays data (HTML)
- The **view** connects the **Model** (data layer) and the **Template** (presentation layer).

Django's Request-Response Cycle

1. Browser Request

- The user enters a URL or clicks a link.
- A **request** is sent to the Django server.

- Example:

  http://127.0.0.1:8000/home/

2. URL Dispatcher (urls.py)

- Django uses urls.py to **match the requested URL** to a view.
- Each URL pattern is mapped to a **specific view**.
- If /home/ is requested, and found in urls.py, Django sends the request to its view.

3. View (views.py)

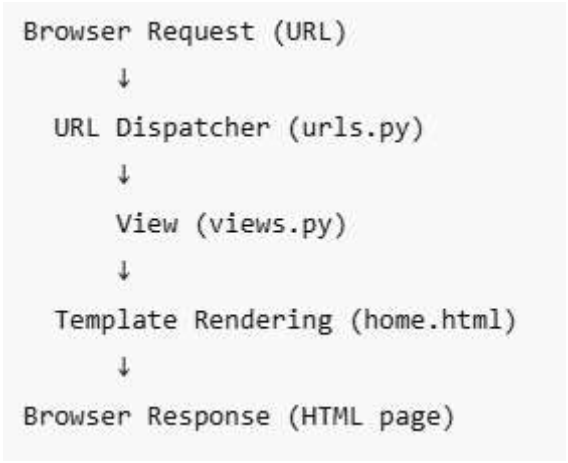
- The view **contains the logic** for the page.
- It may:
  - Fetch data from the database,
  - Prepare information,
  - Choose a template to render.
- It returns a **rendered HTML response** using that template.

4. Template Rendering (home.html)

- The view selects a **template** (usually an HTML file).
- This template is filled with **dynamic data** using Django Template Language (DTL).
- Example: inserting username, product list, etc.

5. Browser Response

- The fully rendered **HTML page** is sent back to the browser.
- The user sees the final web page in their browser window.



Types of Views in Django

1. Function-Based Views (FBVs)

- Most common for beginners
- A simple Python function that takes a request and returns a response
- Uses render(), HttpResponse(), or redirect() to return output

```python
from django.shortcuts import render

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

2. Class-Based Views (CBVs)

- Uses object-oriented programming
- Good for code reuse and modularity
- Django provides built-in CBVs like TemplateView, ListView, DetailView

```python
from django.views.generic import TemplateView

class HomeView(TemplateView):
    template_name = 'home.html'
```

Structure for Views + HTML + URLs :

```
myproject/
├── myapp/
│   ├── views.py        ← Handles logic
│   ├── urls.py         ← Handles routing
│   └── templates/
│       └── myapp/
│           └── home.html  ← Frontend UI
└── myproject/
    └── urls.py         ← Includes all app routes
```

How Views Are Linked to Templates and URLs

1. Linking Views to Templates

- A view tells Django **which template** to display.
- The view prepares the data (if needed) and sends it to the template.
- The template uses Django Template Language (DTL) to display this data as HTML.

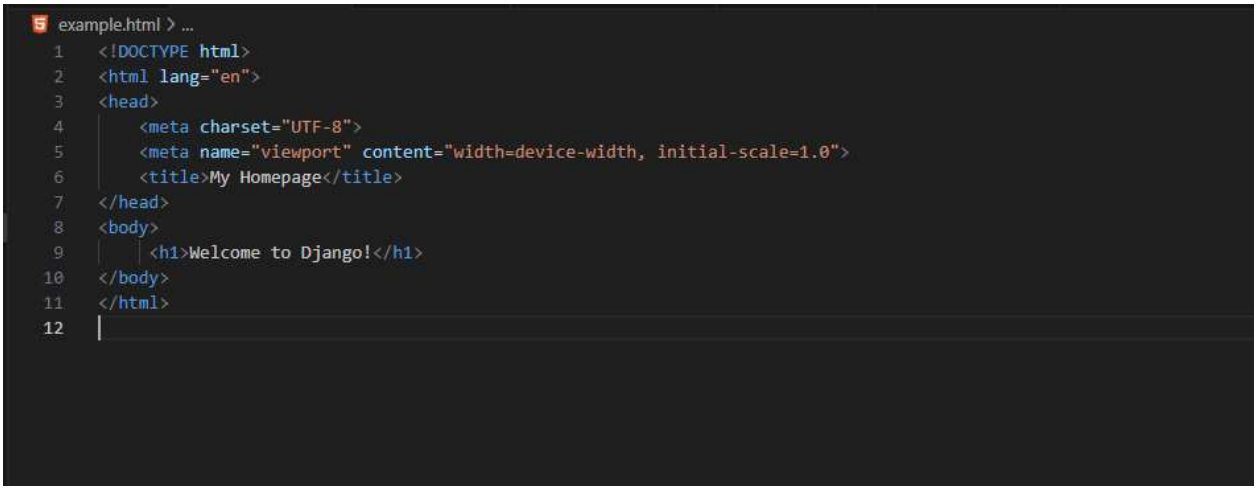
2. Linking Views to URLs

- Each view is connected to a specific **URL pattern** defined in a urls.py file.

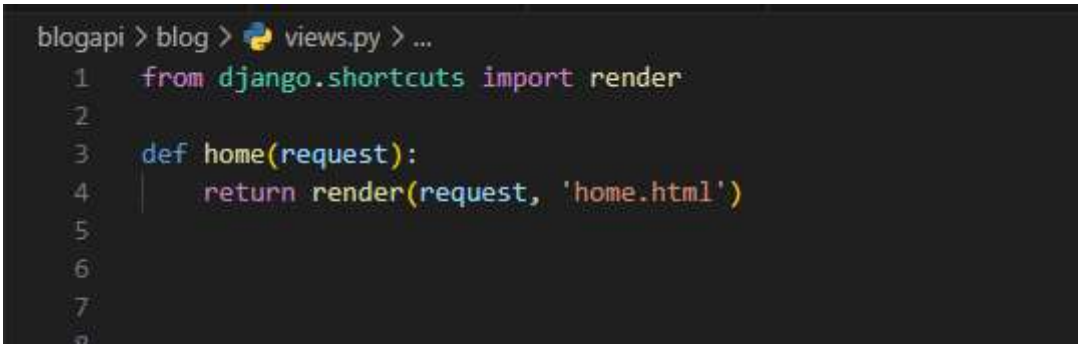
- When a user types a URL in the browser, Django matches it to the correct view using the urls.py routing system.

Example Scenario:

Step 1: Create a Template



Step 2: Create the View



Step 3: Create URL Pattern

blogapi > blog > urls.py > ...

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

urlpatterns = [
    path('', views.home, name='home'),  # Root URL -> home view
]
```

Step 4: Include App URLs in Project URLs

blogapi > blogapi > urls.py > ...

```python
"""
URL configuration for blogapi project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""

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

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