Python

Full stack Skills Bootcamp

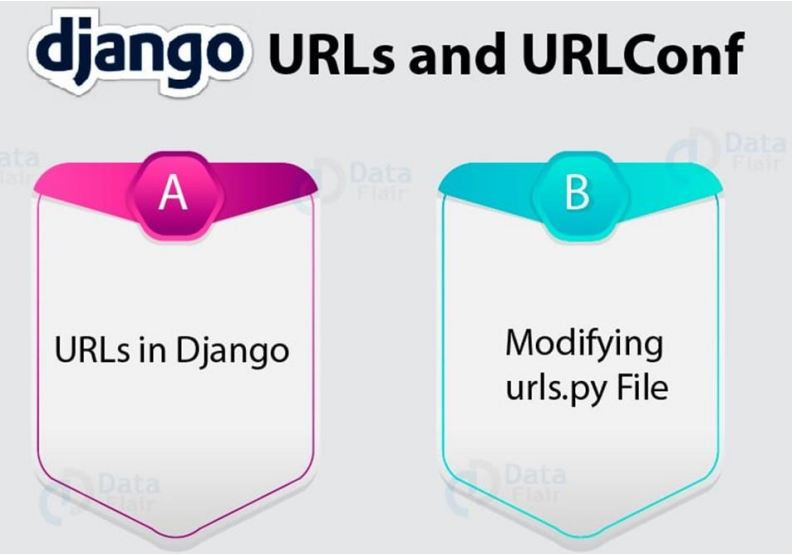
Introduction to Django URLs

What is a URL ?

- URL stands for Uniform Resource Locator, a web address for accessing pages.
- Examples: 127.0.0.1:8000 (local) and https://django.org (online).

Django URLs and URLconf

- Django uses URLconf (URL configuration) to match URLs to specific views.
- Each URL needs its own pattern, so Django knows which view to load.



Understanding Django URL Structure

mysite/urls.py Overview:

mysite/urls.py contains the main URL patterns for the project.

- Docstrings: Triple-quoted lines ("""...""") are for documentation and are ignored by Python.
- Default example in mysite/urls.py:

```python
from django.contrib import admin
from django.urls import path

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

Creating Your First URL Pattern

Adding a Homepage URL

Update mysite/urls.py to include URLs from the blog app:

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

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

- The empty path '' directs http://127.0.0.1:8000/ to blog.urls, where further URL patterns are defined..

Defining URLs in the Blog App

Setting Up blog/urls.py

Create a new file blog/urls.py to handle URLs for the blog app.

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

urlpatterns = [
    path('', views.post_list, name='post_list'),
]
```

- This pattern links the homepage URL to views.post_list. The name='post_list' allows us to refer to this URL later..

Troubleshooting URL Errors

Common URL Error: AttributeError:

- Example Error: AttributeError: module 'blog.views' has no attribute 'post_list'
- Solution: This error appears if post_list is not defined in views.py.
- Fix: Create the post_list view in views.py to resolve this error. Also, restart the server if needed using:

```bash
python manage.py runserver
```

Introduction to Django View

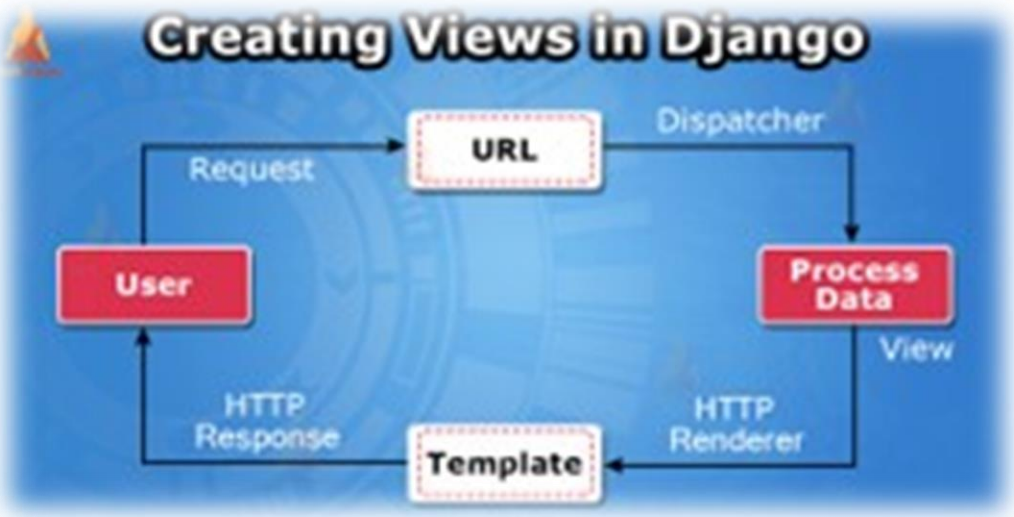
What is a View ?

A view is a Python function or class-based view that takes a **web request** and returns a **web response**.

- It can return HTML pages, JSON data, or even redirect users to other pages.
- Views contain the logic of the application.

```python
from django.http import HttpResponse

def my_view(request):
    return HttpResponse("Hello, World!")
```



Function-Based Views vs Class-Based Views

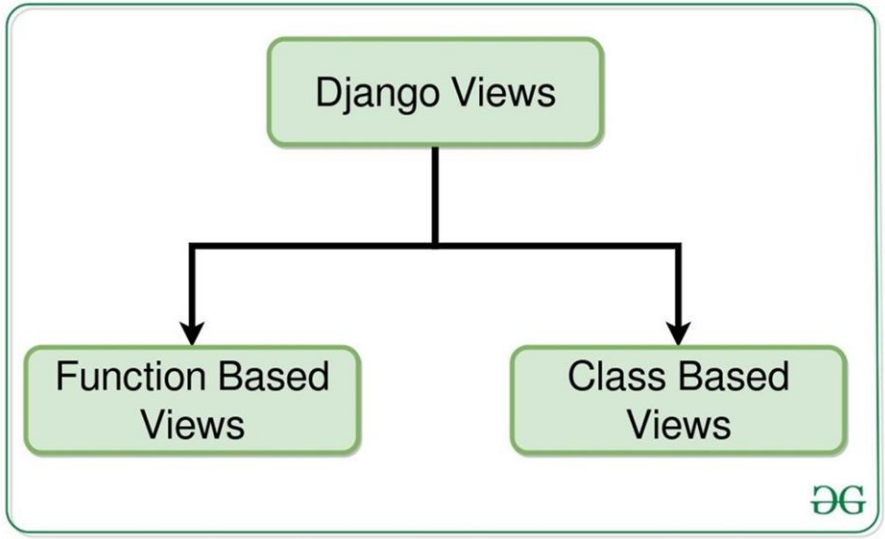
Function-Based Views (FBV): Simple Python functions that take a request and return a response. Ideal for small, simple views with straightforward logic.

```python
def home_view(request):
    return HttpResponse("Home Page")
```

Class-Based Views (CBV): Use Python classes to organize and reuse code. Ideal for complex views that handle multiple actions (e.g., forms, CRUD operations).

```python
from django.views import View
from django.http import HttpResponse

class HomeView(View):
    def get(self, request):
        return HttpResponse("Home Page")
```

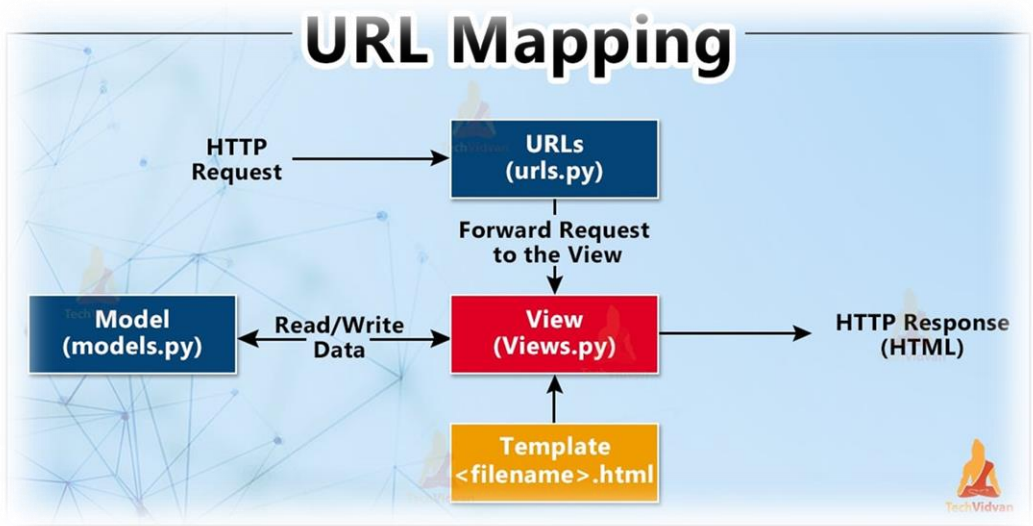


Mapping Views to URLs

- **URLs** direct the browser to the correct view function or class-based view.
- In Django, URLs are mapped to views in the urls.py file.
- Each view (FBV or CBV) has an associated URL pattern.

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

urlpatterns = [
    path('home/', HomeView.as_view()),  # Class-Based View
    path('about/', my_view),  # Function-Based View
]
```



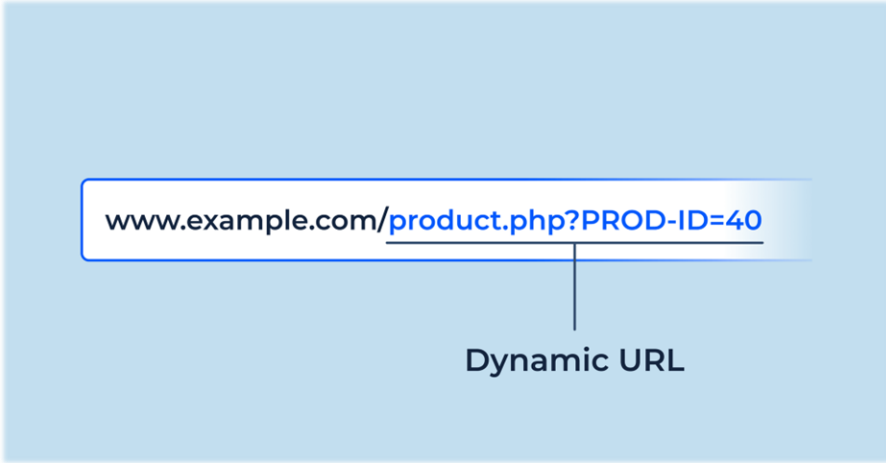
Dynamic URLs in Django

- You can capture data from the URL, such as IDs or usernames, and pass them to views.
- Dynamic URLs allow you to generate flexible paths like /user/1/ or /post/5/.

```python
path('post/<int:id>/', views.post_view),
```

- View function:

```python
def post_view(request, id):
    return HttpResponse(f"Post ID: {id}")
```



URL Namespacing

- Namespaces allow you to organize your URLs, especially in large projects with multiple apps.
- Each app can have its own urls.py file with a namespace for clarity.
- Use the app_name variable to define a namespace and refer to it in templates or views.

```python
app_name = 'blog'

urlpatterns = [
    path('post/<int:id>/', views.post_view, name='post_detail'),
]
```