

Flask

- ↳ USES MVC architecture
- ↓
- Model View Controller → to parts controls.
- ↓
- we write all the utilities (or) code application code the HTML page code.
- 2, used to build small application. (1 page)

3, Non-Monolithic

Django

- ↳ USES MVT architecture
- ↓
- Model View Template
- ↓
- different types of templates can be used.
- 2, used to build scalable applications. (multiple pages)

3, It is monolithic

↳ defined structure of project  
(we need to keep the files in some prescribed structure)

⇒ standard project directory structure.

{  
→ By default we will get admin panel.  
→ we get multiple database connectors.

→ It follows DRY - Don't Repeat Yourself.

① Open Anaconda prompt in Admin mode.

```
Administrator: Anaconda Prompt (anaconda3) - "C:\Users\KrishnaAleti\anaconda3\condabin\conda.bat" activate django
(base) C:\WINDOWS\system32>conda create -n django_env
Collecting package metadata (current_repodata.json): done
Solving environment: done
## Package Plan ##

environment location: C:\Users\KrishnaAleti\anaconda3\envs\django_env
```

→ creating new environment  
to work for django project.  
(conda create -n django-env)

```
Proceed ([y]/n)? y
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
# $ conda activate django_env
#
# To deactivate an active environment, use
# $ conda deactivate

(base) C:\WINDOWS\system32>conda activate django_env
(django_env) C:\WINDOWS\system32>

(base) C:\WINDOWS\system32>conda activate django_env
(django_env) C:\WINDOWS\system32>python -m pip install django
Collecting django
  Using cached Django-4.0.4-py3-none-any.whl (8.0 MB)
Collecting sparse==0.2.2
  Using cached sparse==0.4.2-py3-none-any.whl (42 kB)
Collecting tzdata
  Using cached tzdata-2022.1-py3.py3-none-any.whl (339 kB)
Collecting aspircf<4,>=3.4.1
  Using cached aspircf-3.5.0-py3-none-any.whl (22 kB)
Installing collected packages: tzdata, sparse, aspircf, django
Successfully installed aspircf-3.5.0 django-4.0.4 sparse-0.4.2 tzdata-2022.1
WARNING: You are using pip version 21.1.3; however, version 22.0.4 is available.
You should consider upgrading via the 'C:\Users\KrishnaAleti\AppData\Local\Programs\Python\Python39\python.exe -m pip install --upgrade pip' command.
```

③ activating the newly created environment

④ installing django  
(python -m pip install django)

```
(env_django) C:\WINDOWS\system32>mkdir project
(env_django) C:\WINDOWS\system32>cd project
(env_django) C:\Windows\System32\project>django-admin startproject django_project
(env_django) C:\Windows\System32\project>dir
Volume in drive C is Windows-SSD
Volume Serial Number is 0CF6-84EF

Directory of C:\Windows\System32\project

11-04-2022 04:04 PM <DIR> .
11-04-2022 04:04 PM <DIR> ..
11-04-2022 04:04 PM <DIR> django_project
    0 File(s) 0 bytes
    3 Dir(s) 161,304,285,184 bytes free

```

To create a project.  
→ (5) django-admin startproject project-name

→ project created.

```
(base) C:\Windows\System32\project>D:
(base) D:\>cd D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django
(base) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>conda create -n dj_env
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\KrishnaAleti\anaconda3\envs\dj_env

Proceed {{y/n}}? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
# $ conda activate dj_env
#
# To deactivate an active environment, use
# $ conda deactivate

(base) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>conda activate dj_env
(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>install django
'install' is not recognized as an internal or external command,
operable program or batch file.

(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>python -m pip install django
Requirement already satisfied: django in c:\users\krishnaaleti\appdata\local\programs\python\python39\lib\site-packages (4.0.4)
Requirement already satisfied: tzdata in c:\users\krishnaaleti\appdata\local\programs\python\python39\lib\site-packages (from django) (2022.1)
Requirement already satisfied: asgiref<4,>=3.4.1 in c:\users\krishnaaleti\appdata\local\programs\python\python39\lib\site-packages (from django) (3.5.0)
Requirement already satisfied: sqlparse<0.2.2 in c:\users\krishnaaleti\appdata\local\programs\python\python39\lib\site-packages (from django) (0.4.2)
WARNING: You are using pip version 21.1.3; however, version 22.0.4 is available.
You should consider upgrading via the 'C:\Users\KrishnaAleti\AppData\Local\Programs\Python\Python39\python.exe -m pip install --upgrade pip' command.

(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>
```

```
(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>django-admin startproject django_project
(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>dir
The syntax of the command is incorrect.
```

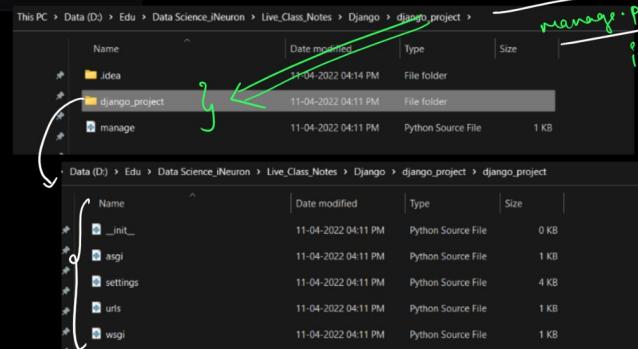
↓  
project created.

⇒ (django-admin startproject project-name)

```
(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>mkd
```

with  
project-name will be created.

→ This will have other folder with name  
same as project name &  
in it.



(6) to run the project (Python manage.py runserver)

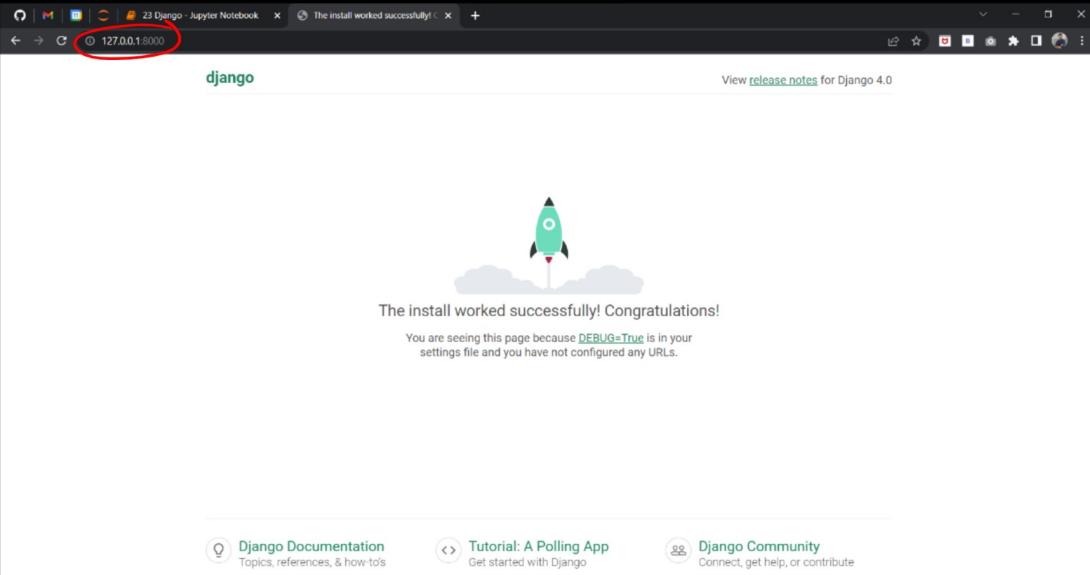
```
(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>python manage.py server
python: can't open file 'D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django\manage.py': [Errno 2] No such file or directory → Error
```

Enter into the project folder & then run the server/project.

```
(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\Django>python manage.py runserver
Matching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
April 11, 2022 - 16:19:25
Django version 4.0.4, using settings 'django_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Start Server ↵



Let's look at what `startproject` created

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

These files are:

- The outer `mysite`/root directory is a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.
  - `manage.py`: A command-line utility that lets you interact with this Django project in various ways. You can read all the details about `manage.py` in [django-admin](#) and `manage.py`.
  - The inner `mysite`/directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. `mysite.urls`).
  - `mysite/__init__.py`: An empty file that tells Python that this directory should be considered a Python package. If you're a Python beginner, read [more about packages](#) in the official Python docs.
  - `mysite/settings.py`: Settings/configuration for this Django project. Django settings will tell you all about how settings work.
  - `mysite/urls.py`: The URL declarations for this Django project; a "table of contents" of your Django-powered site. You can read more about URLs in [URL dispatcher](#).
  - `mysite/asgi.py`: An entry-point for ASGI-compatible web servers to serve your project. See [How to deploy with ASGI](#) for more details.
  - `mysite/wsgi.py`: An entry-point for WSGI-compatible web servers to serve your project. See [How to deploy with WSGI](#) for more details.



The screenshot shows the PyCharm IDE interface with the Django project structure on the left and the `settings.py` file content on the right.

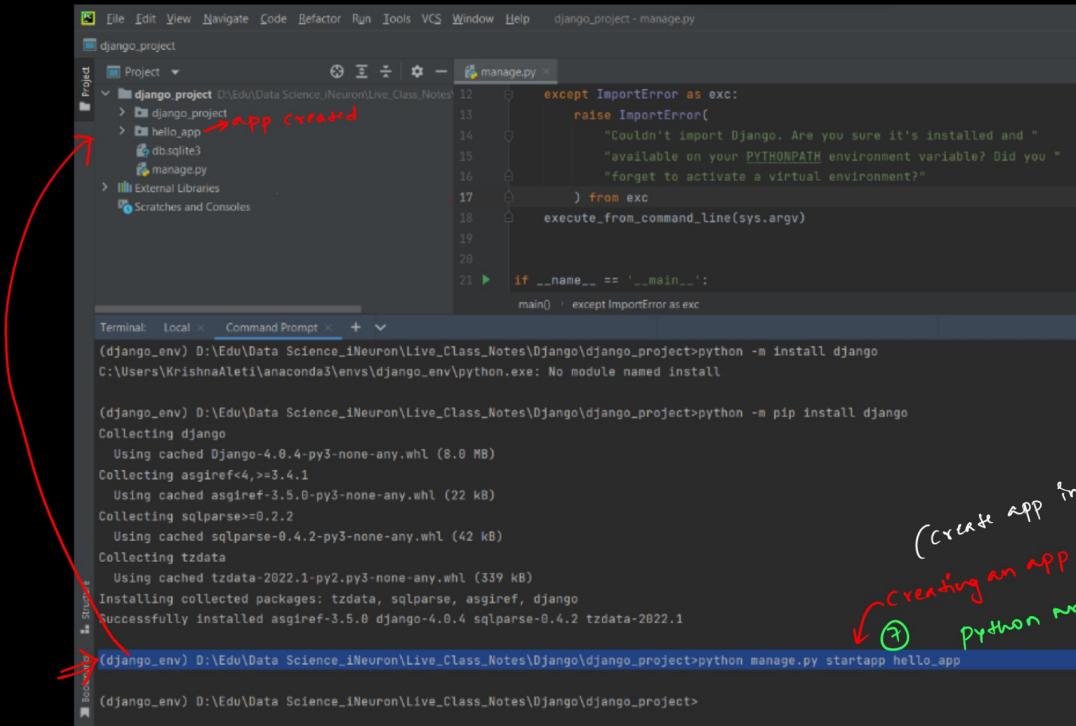
**Project Tree:**

- django\_project** (Project)
- django\_project** (Module)
  - `__init__.py`
  - `asgi.py`
  - `settings.py`** (highlighted with a red arrow)
  - `urls.py`
  - `wsgi.py`
- hello\_world** (Module)
  - `migrations`
  - `__init__.py`
  - `admin.py`
  - `apps.py`
  - `models.py`
  - `tests.py`
  - `urls.py`
  - `views.py`

**Code Editor (settings.py):**

```
75 # Database
76 # https://docs.djangoproject.com/en/4.0/ref/settings/#databases
77
78 DATABASES = {
79     'default': {
80         'ENGINE': 'django.db.backends.sqlite3',
81         'NAME': BASE_DIR / 'db.sqlite3',
82     }
83
84 # Password validation
85 # https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators
86
87 AUTH_PASSWORD_VALIDATORS = [
88     {
89 
```

A red arrow points to the `settings.py` file in the project tree. A red circle highlights the `db.sqlite3` database entry in the code editor.



Name	Date modified	Type	Size
idea	11-04-2022 07:09 PM	File folder	
django_project	11-04-2022 04:19 PM	File folder	
hello_app	11-04-2022 07:13 PM	File folder	
db.sqlite3	11-04-2022 04:19 PM	SQlite3 File	0 KB
manage.py	11-04-2022 04:11 PM	Python Source File	1 KB

hello\_app > views.py

```

from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def index(request):
    text = "<h1>Hello World!</h1>"
    return HttpResponse(text)

```

views.py  
Contains the program logics.

view is to show the info/text in browser.  
(Initially views.py will be empty)  
① we need to write whatever needs to be displayed in the browser)

⇒ To call the view, we need to map it to a URL - and for this we need a URLconf.

To create a URLconf in the hello\_app directory, create a file called urls.py and paste code written as below:-

```

from django.urls import path
from . import views
urlpatterns=[
    path('', views.index, name='index'),
]

```

This means  
from the current directory  
import views.

' ' means  
homepage

⇒ when we hit homepage → views.index will be called

The single dot is a convention from command line applications. It means the current directory. In terms of Django it stands for the directory/module the current file is on

- If the import module in the same dir, use e.g: `from . import core`
- If the import module in the top dir, use e.g: `from .. import core`
- If the import module in the other subdir, use e.g: `from ...other import core`

Note: Starting with Python 2.5, in addition to the implicit relative imports, you can write explicit relative imports with the from module import name form of import statement. These explicit relative imports use leading dots to indicate the current and parent packages involved in the relative import. From the surround module.

project structure:

```

# This django_project.settings file is the one which django is looking for
# for settings, you can define another file as well or you can change the
# file as per different environments different settings file.

# The project settings are
# defined in the manage.py file.

if __name__ == "__main__":
    execute_from_command_line(sys.argv)

```

project settings are defined in the manage.py file.

⇒ Here in the highlighted section, we can define the all the applications of the entire project, so that django will route to the apps defined in this urlpatterns (urls.py of the project)

The next step is to point the root URLconf at the django\_project.urls main module. In django\_project/urls.py, add an import for django.urls.include and insert an include() in the urlpatterns list, so django will look into hello\_world application and in that urls we have index defined so it is able to call that view in this way:

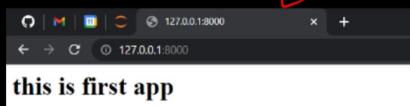
```

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('', include('hello_world.urls')),
    path('admin/', admin.site.urls),
]

```

⇒ run the server → python manage.py runserver



★ ★ Steps to create a project > create an app > run the app:

1, open Anaconda prompt in Admin mode (can use another IDE - but in admin mode)

2, Create a virtual environment for the django project → Conda create -n django-env

↑  
env.name

3, activate the newly created env. → Conda activate django-env

4, Installing django in the virtual env. → python -m pip install django

5, create a project → django-admin startproject django-project

↑  
project name

→ A folder with the project-name will be created.

→ A folder with the project-name & manage.py will be created.

↳ Inside it another folder with project-name & manage.py is there and run server

6, we need to enter into the folder in which manage.py is there and run server  
(outer)

→ python manage.py runserver

→ Open the browser & enter the server url ⇒ http://127.0.0.1:8000/

⇒ A page with 'The install worked successfully! Congratulations!' will be displayed.

7, Now create our own app in the django-project folder → python manage.py startapp hello-world

↑  
app name

→ The app (hello-world|hello-app) gets created successfully.

8, In the views.py (of the app: hello-world), we need to write whatever we want to display in the browser.

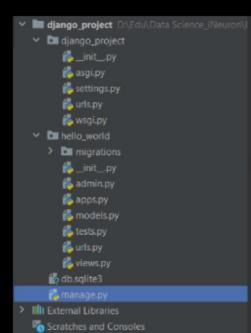
```

from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.

def index(request):
    text = "<h1> Hey There!!! This is Krish </h1>"
    return HttpResponse(text)

```



f9  
structure  
(we always  
need to  
follow this  
structure only)

→ Now to call this view, we need to map it to a url - and for this we need a URLconfig.

9, To create URLconfig in the hello\_world directory, create a file called urls.py & write the following code:

```
django_project / hello_world / urls.py
Project / django_project / urls.py
  / django_project
    / __init__.py
    / asgi.py
    / settings.py
    / urls.py
    / wsgi.py
  / hello_world
    / migrations
      / __init__.py
      / admin.py
      / apps.py
      / models.py
      / tests.py
      / urls.py
      / views.py
    / db.sqlite3
    / manage.py
```

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

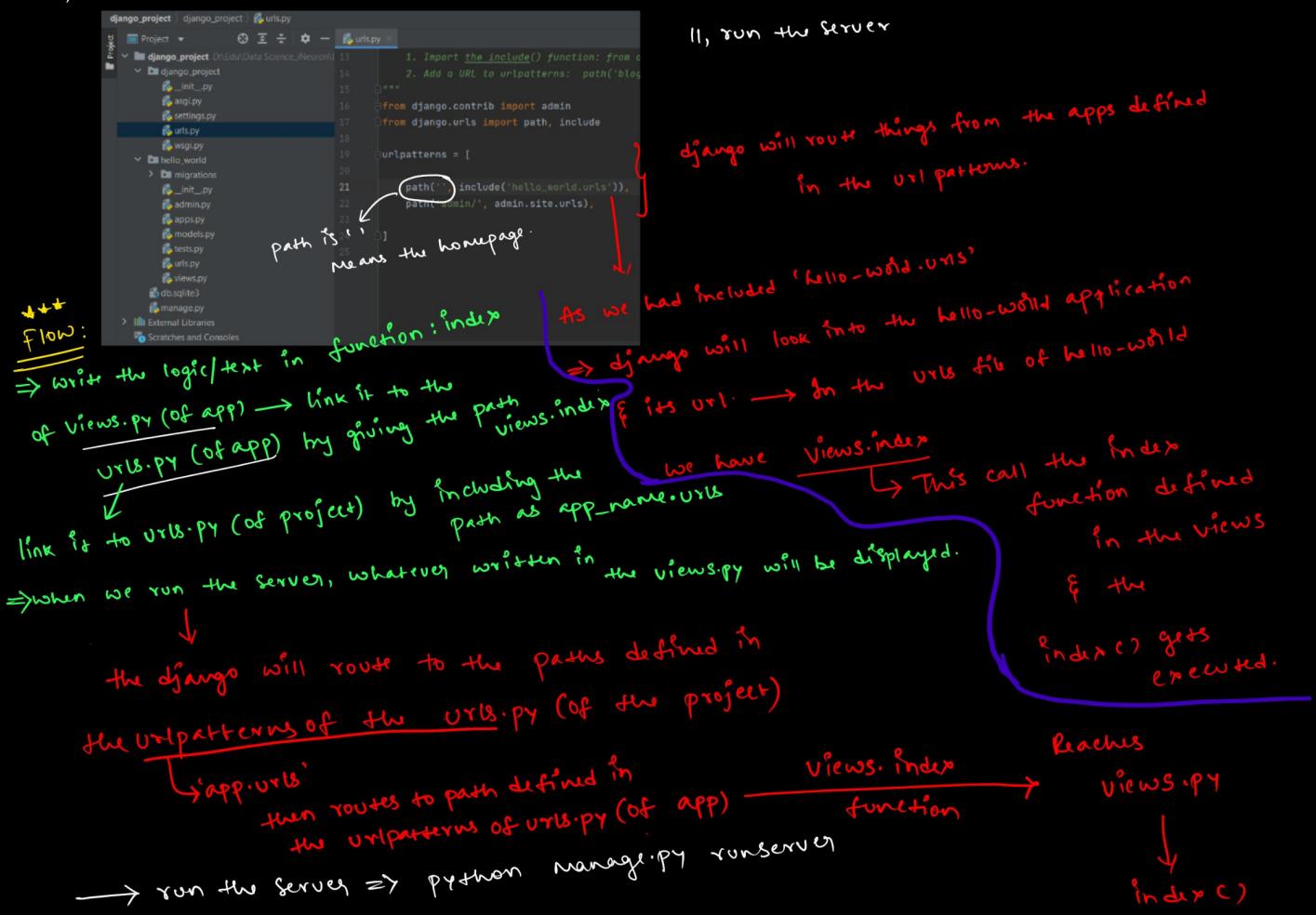
urlpatterns = [
    path('', views.index, name='index'),
]
```

In views file, we had defined the 'index' function.

10, The next step is to point the root URLconfig at the django-project.urls main module.

11, run the server

django will route things from the apps defined in the url patterns.



→ Whatever written in the function will be displayed in the browser.

## Django file linking / flow:

### views.py of app

```

views.py
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4
5 def index(request):
6     text = "<h1> Hey There!!! This is Krish </h1>"
7     return HttpResponse(text)
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

function/ logic

Create your views here.

we can call this 'index'

in other modules as

views.index()

because the index() function

is in views.py file.

### urls.py of app

```

urls.py of app
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path('', views.index, name='index'),
7     1
8 ]

```

### urls.py of project

```

urls.py of project
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

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'))

```

urls.py of project
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

path('', include('hello\_world.urls')),  
path('admin/', admin.site.urls))

⇒ when we run the Server - python manage.py runserver

the django looks into the project's urls.py file & routes to the paths/urls defined in the urlpatterns of urls.py (of project)

⇒ The flow is:

we run server → django looks into urls.py of project

```

views.py x hello_world.urls.py x django_projecturls.py
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

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'))

```

urls.py x hello_world.urls.py x django_projecturls.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

path('', include('hello\_world.urls')),  
path('admin/', admin.site.urls))

```

views.py x hello_world.urls.py x django_projecturls.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

from django.urls import path

from . import views

urlpatterns = [

path('', views.index, name='index'),

1

```

views.py x hello_world.urls.py x django_projecturls.py
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

# Create your views here.

def index(request):

text = "<h1> Hey There!!! This is Krish </h1>"

return HttpResponse(text)

we get the result.

run server

django looks in to  
urls.py (of project)  
i.e., the urlpatterns

urlpatterns = app.urls  
(urls.py of project)

looks into  
urls.py  
(urlpatterns)  
of app

django looks into  
the

views.py

⇒ and executes the index function

Returns the result.

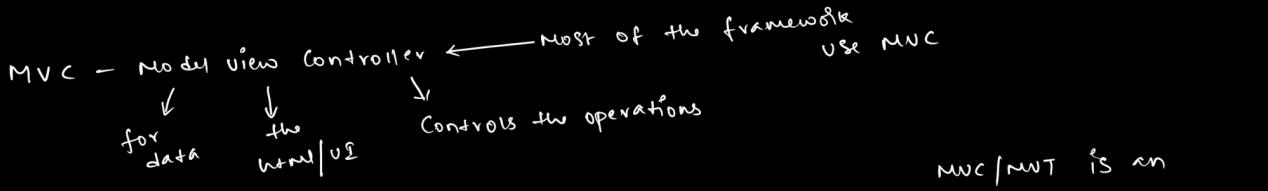


# Django (by Telusko)

Django - Free and open source web framework  
 ↴  
 used to build web applications

Combination of Components & packages  
 ↓  
 Using 'Python' for backend.

⇒ We can use Django as framework to build web applications



⇒ Django follows MVT - Model View Template

Controller of MVC = View of MVT

View of MVC = Template of MVT

## Django Setup:

Django is a web framework for python

→ Open Anaconda prompt in admin mode

→ Create a virtual environment

(Conda create -n django-telusko)

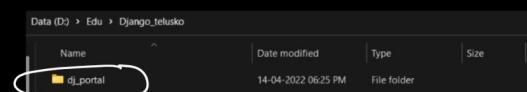
→ activate the virtual environment &

install django: pip install django

→ Create a project = dj-portal

① django-admin startproject projectname

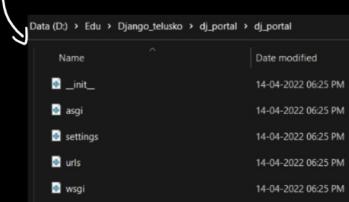
⇒ A project will be created (folder with projectname)



Inside it another folder

with same (project) name &

manage.py will be present



→ we need to enter the project folder (outer)

```

(django_telusko) D:\Edu\ Django_telusko>django-admin startproject dj_portal
(django_telusko) D:\Edu\ Django_telusko>cd dj_portal
(django_telusko) D:\Edu\ Django_telusko\dj_portal>dir
Volume in drive D is Data
Volume Serial Number is B49E-AEJF
Directory of D:\Edu\ Django_telusko\dj_portal

14-04-2022 06:25 PM <DIR>
14-04-2022 06:25 PM <DIR> ..
14-04-2022 06:25 PM <DIR> dj_portal }
14-04-2022 06:25 PM 687 manage.py
1 File(s) 687 bytes
3 Dir(s) 944,671,453,184 bytes free
(django_telusko) D:\Edu\ Django_telusko\dj_portal>
  
```

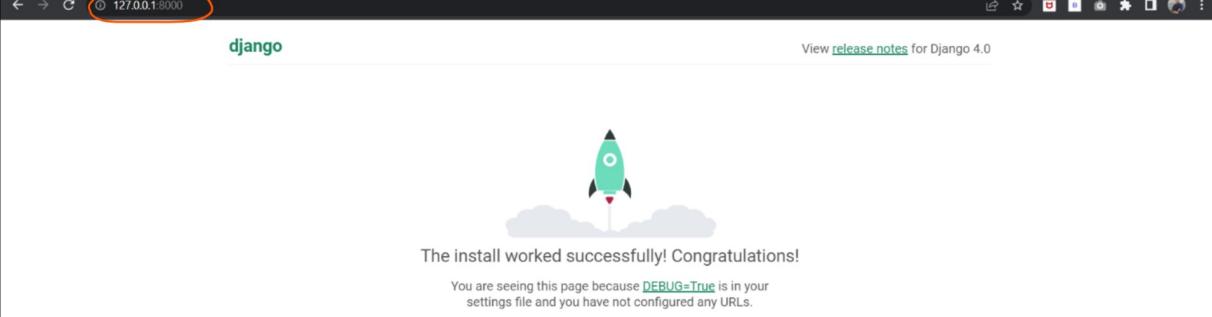
→ django provides a server to us.

## ② python manage.py runserver

```
(django_telusko) D:\Edu\ Django_telusko\ dj_portal>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
April 14, 2022 - 18:34:07
Django version 4.0.4, using settings 'dj_portal.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

✓ Server → http://127.0.0.1:8000/  
local host      port number



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

## Creating App in Django:

Open IDE (PyCharm) in admin mode.

In django project we will have multiple apps.

like Amazon - it has Cart, wishlist etc..., so if we want to build Amazon project, we need to create different apps for different

modules (Cart, wishlist etc...)

→ Create an app ⇒ A folder with app name (calc) will be created.

③ → `python manage.py startapp calc`  
↓  
app name

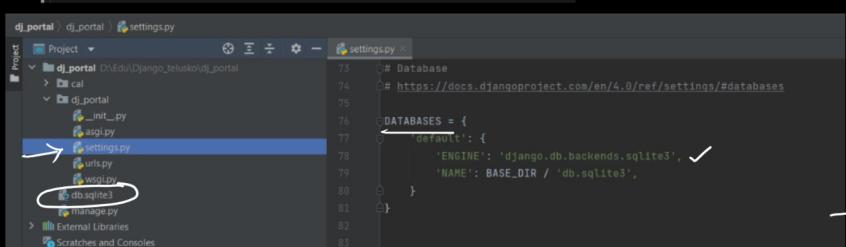
```
(django_telusko) PS D:\Edu\ Django_telusko\ dj_portal> python manage.py startapp calc
```

Name	Date modified	Type	Size
idea	14-04-2022 06:55 PM	File folder	
cal	14-04-2022 07:25 PM	File folder	
dj_portal	14-04-2022 06:34 PM	File folder	
db.sqlite3	14-04-2022 06:34 PM	SQlite3 File	0 KB
manage	14-04-2022 06:25 PM	Python Source File	1 KB

Project	D:\Edu\ Django_telusko\ dj_portal
	cal └── migrations └── __init__.py └── admin.py └── apps.py └── models.py └── tests.py └── views.py dj_portal └── __init__.py └── asgi.py └── settings.py └── urls.py └── wsgi.py db.sqlite3 manage.py

Name	Date modified	Type	Size
migrations	14-04-2022 07:25 PM	File folder	
__init__	14-04-2022 07:25 PM	Python Source File	0 KB
admin	14-04-2022 07:25 PM	Python Source File	1 KB
apps	14-04-2022 07:25 PM	Python Source File	1 KB
models	14-04-2022 07:25 PM	Python Source File	1 KB
tests	14-04-2022 07:25 PM	Python Source File	1 KB
views	14-04-2022 07:25 PM	Python Source File	1 KB

```
dj.portal> dj_portal> settings.py
```



```
73 # Database
74 # https://docs.djangoproject.com/en/4.0/ref/settings/#databases
75
76 DATABASES = {
77     'default': {
78         'ENGINE': 'django.db.backends.sqlite3',
79         'NAME': BASE_DIR / 'db.sqlite3',
80     }
81 }
```

→ When we create a project, a `settings.py` will also be created & it will have imp. info.  
→ In it, the database is `sqlite3` by default, hence a `db.sqlite3` file will be created in the project.

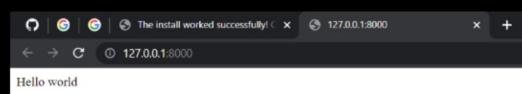
④ Write functions & URLs in `views.py`

```
views.py x cal.urls.py x dj_portal\urls.py x
1 from django.shortcuts import render, HttpResponseRedirect
2 from django.http import HttpResponse
3
4 # Create your views here.
5
6 def home(request):
7     return HttpResponseRedirect("Hello world")
```

```
views.py x cal.urls.py x dj_portal\urls.py x
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.home, name='home'),
6 ]
```

```
views.py x cal.urls.py x dj_portal\urls.py x
1 Class-based view
2
3     1. Add an import: from other_app.views import Home
4     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
5 Including another URLconf
6     1. Import the include() function: from django.urls import include, path
7     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
8
9 urlpatterns = [
10     path('', include('cal.urls')),
11     path('admin/', admin.site.urls),
12 ]
```

⑤ `python manage.py runserver` → Result:



## Django Template Language (DTL):

to display the dynamic content

used for printing Dynamic Content which means  
⇒ Not directly writing the text in return statement  
but writing the text content in home file & giving the file in return statement.

The screenshot shows the Visual Studio Code interface with the project structure on the left. The terminal at the bottom shows the command `python manage.py runserver` being run, and the output shows the server is running on port 8000.

```
File Edit Selection View Go Run Terminal Help
EXPLORER
DJ PORTAL
> .idea
> cal
> __pycache__
> migrations
> __init__.py
> admin.py
> apps.py
> models.py
> tests.py
> urls.py
> views.py
dj_portal
> __pycache__
> __init__.py
> asgi.py
> settings.py
> urls.py
> wsgi.py
db.sqlite3
env.info.txt
manage.py

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.
D:\Edu\ Django\ telusko\ dj_portal> conda activate django_telusko
(base) D:\Edu\ Django\ telusko\ dj_portal> conda activate django_telusko
(django_telusko) D:\Edu\ Django\ telusko\ dj_portal> dir
Volume in drive D is Data
Volume Serial Number is B49E-AE1F
Directory of D:\Edu\ Django\ telusko\ dj_portal

14-04-2022 09:31 PM <DIR> .
14-04-2022 09:25 PM <DIR> ..
14-04-2022 09:38 PM <DIR> .idea
14-04-2022 09:58 PM <DIR> cal
14-04-2022 09:34 PM 0 db.sqlite3
```

In templates, we will have HTML files.

→ The layout may be fixed but the content can be made dynamic.

① Create 'templates' folder & add .html file in it.

Create a folder: 'templates' in the project folder & create a (outer)

HTML file in the templates.

```
views.py x home.html x settings.py x urls.py cal x urls.py dj_portal x
templates > home.html
1 <h1> Hello world, This is Krish </h1>
```

Mention the text to be displayed in `home.html` when it is called.

⇒ In 'templates' folder, we keep our HTML files.

```

EXPLORER
DJ_PORTAL
> idea
> cal
dj_portal
> __pycache__
< settings.py
< urls.py
< wsgi.py
templates
> home.html
db.sqlite3
env.info.txt
manage.py

views.py  o home.html  settings.py  urls.py cal  urls.py dj_portal
51
52 ROOT_URLCONF = 'dj_portal.urls'
53
54 TEMPLATES = [
55     {
56         'BACKEND': 'django.template.backends.django.DjangoTemplates',
57         'DIRS': [], ✓
58         'APP_DIRS': True,
59         'OPTIONS': {
60             'context_processors': [
61                 'django.template.context_processors.debug',
62                 'django.template.context_processors.request',
63                 'django.contrib.auth.context_processors.auth',
64                 'django.contrib.messages.context_processors.messages',
65             ],
66         },
67     },
68 ],
69
70 WSGI_APPLICATION = 'dj_portal.wsgi.application'

```

(2) Add the created 'templates' folder path in the 'DIRS' of the 'Templates' in the settings.py of the project.

In the settings.py, in Templates, we need to mention the directory where our templates folder is present.

so that the django can look into the files present in our 'templates' folder

```

File Edit Selection View Go Run Terminal Help
EXPLORER
DJ_PORTAL
> idea
> cal
dj_portal
> __pycache__
< settings.py
< urls.py
< wsgi.py
templates
> home.html
db.sqlite3
env.info.txt
manage.py

views.py  o home.html  settings.py  urls.py cal  urls.py dj_portal
51
52 ROOT_URLCONF = 'dj_portal.urls'
53
54 TEMPLATES = [
55     {
56         'BACKEND': 'django.template.backends.django.DjangoTemplates',
57         'DIRS': [os.path.join(BASE_DIR, 'templates')], ✓
58         'APP_DIRS': True,
59         'OPTIONS': {
60             'context_processors': [
61                 'django.template.context_processors.debug',
62                 'django.template.context_processors.request',
63                 'django.contrib.auth.context_processors.auth',
64                 'django.contrib.messages.context_processors.messages',
65             ],
66         },
67     },
68 ],
69
70 WSGI_APPLICATION = 'dj_portal.wsgi.application'

```

Giving the directory of our 'templates' folder.

we need to render our template (.html file) because template is a file & it can have some dynamic content. so, we have to merge the static content with dynamic content and that is called 'rendering'.

(3) In views.py call the .html file

```

cal > views.py > o home
1 from django.shortcuts import render, HttpResponseRedirect
2 from django.http import HttpResponseRedirect
3
4
5 # Create your views here.
6
7 def home(request):
8     # return HttpResponseRedirect("Hello world")
9     return render (request, 'home.html')
10

```

↓  
.html file name

(4) runserver



```

File Edit Selection View Go Run Terminal Help
EXPLORER
DJ_PORTAL
> idea
> cal
dj_portal
> __pycache__
< templates
< home.html
db.sqlite3
env.info.txt
manage.py

views.py  o home.html  settings.py  urls.py cal  urls.py dj_portal
51
52 ROOT_URLCONF = 'dj_portal.urls'
53
54 TEMPLATES = [
55     {
56         'BACKEND': 'django.template.backends.django.DjangoTemplates',
57         'DIRS': [os.path.join(BASE_DIR, 'templates')], ✓
58         'APP_DIRS': True,
59         'OPTIONS': {
60             'context_processors': [
61                 'django.template.context_processors.debug',
62                 'django.template.context_processors.request',
63                 'django.contrib.auth.context_processors.auth',
64                 'django.contrib.messages.context_processors.messages',
65             ],
66         },
67     },
68 ],
69
70 WSGI_APPLICATION = 'dj_portal.wsgi.application'

```

This is the flow for getting the dynamic Content (Code-wise we can make changes)

```

cal > views.py > o home.html  settings.py  urls.py
templates > home.html
1 <h1> Hello world, This is krish </h1>
2
3
4 # Create your views here.
5
6 def home(request):
7     # return HttpResponseRedirect("Hello world")
8     return render (request, 'home.html')
9
10

```

(1)

(3)

Result  
Hello world, This is krish

```

cal > urls.py > ...
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.home, name='home'),
6 ]

```

(2)

(4)

Result  
Hello world, This is krish

## Making the Content more dynamic:

```
views.py
from django.shortcuts import render, HttpResponseRedirect
from django.http import HttpResponseRedirect
# Create your views here.
def home(request):
    # return HttpResponseRedirect("Hello world")
    return render(request, 'home.html', {'name': 'krishna'})
```

we can get this name from anywhere, this make the content more dynamic



→ we are not fixing the content in the file to be displayed, we are fetching the info from other file/code which makes our code dynamic.

info (name) from other file/code which makes our code dynamic.  
↳ (python code - views.py)

```
views.py
from django.shortcuts import render, HttpResponseRedirect
from django.http import HttpResponseRedirect
# Create your views here.
def home(request):
    # return HttpResponseRedirect("Hello world")
    return render(request, 'home.html', {'name': 'krishna'})
```

the curly bracket here means that the code is dynamic,

→ this is "Jinja pattern" (or) "Jinja Template"

→ In the curly brackets, instead of variable name, we can give expressions also.

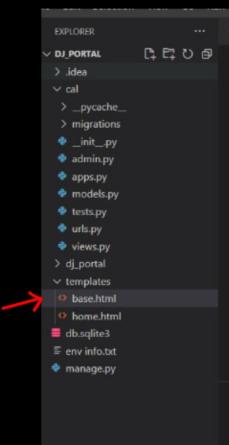
## Making the Content Dynamic - creating base.html:

In the website, we will have multiple pages. The UI can be same but the ↳ (layout, colorscheme etc..)

Content displayed in different pages will be different.

→ So, we can take the content/code which is same for all pages & keep it in a file.

Creating a HTML file (base.html) & keeping basic code which will be common for all the pages.



We kept the common code in base.html, which means the pages will use this base.html & their own code as per the page requirements.

Extending the base.html to home.html

```
base.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Krishna Portal</title>
</head>
<body background="cyan">
    <% block content %>
    <% endblock %>

```

basic HTML code

color for the page

In the body we mention the block content

```
home.html
<% extends 'base.html' %>
<% block content %>
    <h1> Hello world, This is {{name}} </h1>
<% endblock %>
```

Content written in the block content will be sent to 'block content' of base.html

```
views.py
from django.shortcuts import render, HttpResponseRedirect
from django.http import HttpResponseRedirect
# Create your views here.
def home(request):
    # return HttpResponseRedirect("Hello world")
    return render(request, 'home.html', {'name': 'krishna'})
```

base.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie-edge">
7     <title>krish_Dj</title>
8   </head>
9   <body bgcolor='cyan'>
10    {% block content %}</body>
11  </html>
12
13
14
15
16

```

home.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie-edge">
7     <title>krish_Dj</title>
8   </head>
9   <body>
10    <h1>Hello world, This is krishna</h1>
11    <h2>{{ name }}</h2>
12  </body>
13</html>

```

urls.py (dj-portal)

```

1 from django.urls import path
2
3 urlpatterns = [
4     path('add', views.add, name='add'),
5     path('home', views.home, name='home'),
6 ]

```

views.py

```

1 from django.shortcuts import render, HttpResponseRedirect
2 from django.http import HttpResponse
3
4 def home(request):
5     return render(request, 'home.html', {'name': 'krishna'})
6
7 def add(request):
8     if request.method == 'POST':
9         num1 = request.POST['num1']
10        num2 = request.POST['num2']
11        result = int(num1) + int(num2)
12        return render(request, 'result.html', {'result': result})
13    else:
14        return render(request, 'result.html')

```

Django Dev Server

Hello world, This is krishna

Server → urls.py (dj-portal) → urls.py (cal)

↓

views.py

home.html ↗

base.html + home.html ↘

In home.html, we had written 'extends base.html', which means base.html + home.html

Addition of two numbers in Django

base.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie-edge">
7     <title>Dj app</title>
8   </head>
9   <body bgcolor='cyan'>
10    {% block content %}</body>
11  </html>
12
13
14
15
16

```

home.html

```

1 {% extends 'base.html' %}
2
3 {% block content %}
4
5 <h1>Hello world, This is {{ name }}. Below is the calculator app.</h1>
6
7 <form action = "add">
8   Enter 1st number: <input type="text" name="num1"><br> {# comment %} "br to get in the new line" {# endcomment %}
9   Enter 2nd number: <input type="text" name="num2"><br>
10  <input type="submit">
11 </form>
12
13
14  {% endblock %}

```

Django Dev Server

Hello world, This is krishna. Below is the calculator app.

Enter 1st number:  ↗ form  
Enter 2nd number:   
Submit

Creating a form

After we enter the num1, num2 & click submit, the action = 'add'

Django Dev Server

Hello world, This is krishna. Below is the calculator app.

Enter 1st number: 5  
Enter 2nd number: 4  
Submit

Giving num1, num2 & click on submit.

then the action is 'add', which we had mentioned by not mapped anywhere → so, we will get error.

→ mapping the path for 'add':

Page not found (404)

We need to include the path for 'add' to fix the issue.

Request Method: GET  
Request URL: http://127.0.0.1:8000/add?num1=5&num2=4

Using the URLconf defined in dj\_portal.urls, Django tried these URL patterns, in this order:

1. [name='home']
2. admin/

The current path, add, didn't match any of these.

You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.

base.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie-edge">
7     <title>Dj app</title>
8   </head>
9   <body bgcolor='cyan'>
10    {% block content %}</body>
11  </html>
12
13
14
15
16

```

urls.py (cal)

```

1 from django.urls import path
2
3 urlpatterns = [
4     path('add', views.add, name='add'),
5     path('home', views.home, name='home'),
6 ]

```

views.py

```

1 from django.shortcuts import render, HttpResponseRedirect
2 from django.http import HttpResponse
3
4 def home(request):
5     return render(request, 'home.html', {'name': 'krishna'})
6
7 def add(request):
8     if request.method == 'POST':
9         num1 = request.POST['num1']
10        num2 = request.POST['num2']
11        result = int(num1) + int(num2)
12        return render(request, 'result.html', {'result': result})
13    else:
14        return render(request, 'result.html')

```

result.html

```

1 {% extends 'base.html' %}
2
3 {% block content %}
4
5 <h1>result</h1>
6
7  {% endblock %}

```

Django Dev Server

result

Flow for adding two numbers in Django

Used base.html, home.html, result.html  
 (We need to add our templates path in 'Dirs' of Templates in settings.py)

```

        ① settings.py          ② urls.py dj_portal          ③ urls.py cal          ④ base.html          ⑤ home.html          ⑥ views.py          ⑦ result.html
    
```

for home page

for path 'add'

go to Views.add ⑥

④ This gets printed & also the form will be displayed.

⑤ Once we enter the values & click 'Submit' then the django will look for the form's action to perform which is 'add' => django will look into "urlpatterns" of urls.py (of app) for the path defined for 'add' & routes to the module/function defined in that path.

⑦ data(num1,num2) are displayed.

⑧ because by default we get num1, num2 as strings from the form

```

settings.py
dj.portal > urls.py dj_portal ...
14     2. Add a URL to urlpatterns: path('blog/', include ...
15     ...
16     from django.contrib import admin
17     from django.urls import path, include
18
19     urlpatterns = [
20         path('', include('cal.urls')),
21         path('admin/', admin.site.urls),
22     ]
23

urls.py dj_portal
cal > urls.py > ...
1     from django.shortcuts import render, HttpResponseRedirect
2     from django.http import HttpResponseRedirect
3
4
5     # Create your views here.
6
7     def home(request):
8         # return HttpResponseRedirect("Hello world")
9         return render (request, 'home.html', {'name':'krishna'})
10
11    def add(request):
12        num1 = request.GET['num1']
13        num2 = request.GET['num2']
14        res = int(num1) + int(num2)
15        return render(request, 'result.html', {'result':res})
16

views.py
cal > views.py > ...
1     from django.shortcuts import render, HttpResponseRedirect
2     from django.http import HttpResponseRedirect
3
4
5     # Create your views here.
6
7     def home(request):
8         # return HttpResponseRedirect("Hello world")
9         return render (request, 'home.html', {'name':'krishna'})
10
11    def add(request):
12        num1 = request.GET['num1']
13        num2 = request.GET['num2']
14        res = int(num1) + int(num2)
15        return render(request, 'result.html', {'result':res})
16

base.html
templates > base.html
1     <!DOCTYPE html>
2     <html lang="en">
3         <head>
4             <meta charset="UTF-8">
5             <meta name="viewport" content="width=device-width, initial-scale=1.0">
6             <title>Dj_app</title>
7         </head>
8         <body bgcolor="cyan">
9             {% block content %}
10            ...
11        {% endblock %}
12    </body>
13 </html>
14

home.html
templates > home.html
1     {% extends 'base.html' %}
2
3     {% block content %}
4
5         <h1> Hello world, This is {{name}}. Below is the calculator app. </h1>
6
7         <form action = "add">
8             Enter 1st number: <input type="text" name="num1"> <br> (% comment %) "br" to get in the new line (% endcomment %)
9             Enter 2nd number: <input type="text" name="num2"> <br>
10            <input type="submit">
11        </form>
12
13    {% endblock %}
14

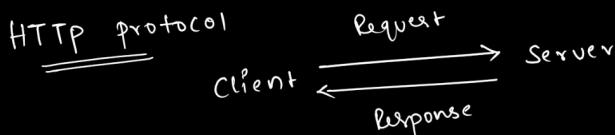
result.html
templates > result.html
1     {% extends 'base.html' %}
2
3     {% block content %}
4
5         result: {{result}}
6
7    {% endblock %}
18

Django shell
127.0.0.1:8000
Hello world, This is krishna. Below is the calculator app.
Enter 1st number: 5
Enter 2nd number: 6
Submit
result: 11
127.0.0.1:8000/add?num1=5&num2=6

```

# Django (Tewsko)

## GET Vs POST - HTTP methods



→ when we want to fetch resource from the server, we use GET Request.



by sending some data

(we are expecting something from the server)

→ POST - sending something to server  
(like filling a form & submitting it)

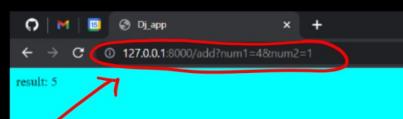
(Adding data to server)



→ PUT - updating data on the server

```

settings.py      urls.py dj_portal      urls.py cal      views.py x      result
al > * views.py > ...
1  from django.shortcuts import render, HttpResponseRedirect
2  from django.http import HttpResponseRedirect
3
4
5  # Create your views here.
6
7  def home(request):
8      # return HttpResponseRedirect("Hello world")
9      return render(request, 'home.html', {'name': 'krishna'})
10
11 def add(request):
12     val1 = request.GET['num1']
13     val2 = request.GET['num2']
14     res = int(val1) + int(val2)
15     return render(request, 'result.html', {'result': res})
16
  
```



We had used 'GET' method in our code for the addition of the 2 numbers.

→ In GET method, we send the data (num1, num2) in the address bar.

⇒ In this case our data will be visible to everyone & is not secured.

(for eg.. login credentials)

→ so to overcome this we use 'POST' method.

```

settings.py      urls.py dj_portal      urls.py cal      views.py      result.html x      base.html
templates > * home.html
1  (% extends 'base.html' %)
2
3  (% block content %)
4
5  <h1> Hello world, This is {{name}}. Below is the calculator app. </h1>
6
7  <form action = "add", method='POST'> → In the form, mention method='POST'
8      Enter 1st number: <input type="text" name="num1"><br> [% comment %] "br" to get in the next line [& endcomment %]
9      Enter 2nd number: <input type="text" name="num2"><br>
10     <input type="submit">
11
12 </form>
13
14 (% endblock %)
15
16
  
```

The entire code of 'Addition of two numbers in Django' will be same. We just need to change the method from GET to POST.

Hello world, This is krishna. Below is the calculator app.

Enter 1st number:

Enter 2nd number:

we got error but the values are not displayed in the url.

Forbidden (403)

CSRF verification failed. Request aborted.

You are seeing this message because this site requires a CSRF cookie when submitting forms. This cookie is required for security reasons, to ensure that your browser is not being hijacked by third parties.

If you have configured your browser to disable cookies, please re-enable them, at least for this site, or for "same-origin" requests.

Help

Reason given for failure:  
CSRF cookie not set.

In general, this can occur when there is a genuine Cross Site Request Forgery, or when Django's CSRF mechanism has not been used correctly. For POST forms, you need to ensure:

- Your browser is accepting cookies.
- The view function passes a request to the template's render method.
- In the template, there is a `{% csrf_token %}` template tag inside each POST form that targets an internal URL.
- If you are not using `csrf_protect`, then you must use `csrf_exempt` on any views that use the `{% csrf_token %}` template tag, as well as those that accept the POST data.
- The form has a valid CSRF token. After logging in in another browser tab or hitting the back button after a login, you may need to reload the page with the form, because the token is rotated after a login.

You're seeing the help section of this page because you have `DEBUG = True` in your Django settings file. Change that to `False`, and only the initial error message will be displayed.

You can customize this page using the `CSRF_FAILURE_VIEW` setting.

Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. CSRF attacks exploit the trust a Web application has in an authenticated user.

so, to avoid this insecurity, In django we have middlewares.  
we need to use them.

```
settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware', ✓
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

home.html

```
<h1>Hello world, This is {{name}}. Below is the calculator app. </h1>
<form action = "add", method="POST">
    [% csrf_token %]
    Enter 1st number: <input type="text" name="num1"><br> {# comment #} br to get in the new line" {# endcomment #}
    Enter 2nd number: <input type="text" name="num2"><br>
    <input type="submit">
</form>
</div>
```

so, we are using csrf-token to send the data now.

```
views.py
def home(request):
    # return HttpResponseRedirect("Hello world")
    return render(request, 'home.html', {'name': 'krishna'})

def add(request):
    val1 = request.POST['num1']
    val2 = request.POST['num2']
    res = int(val1) + int(val2)
    return render(request, 'result.html', {'result': res})
```

NO data in the url [address bar]

result: 12

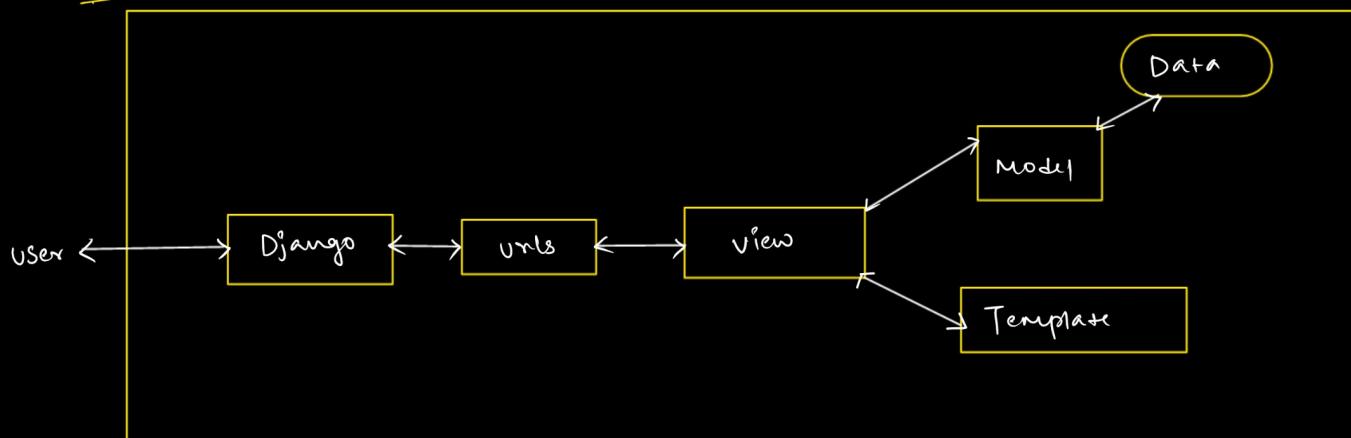
## Model View Template MVT / MTV

→ In Django, we used MVT.

↳ Model View Template

→ In every web app we need - layout, logic, data

MVT framework:



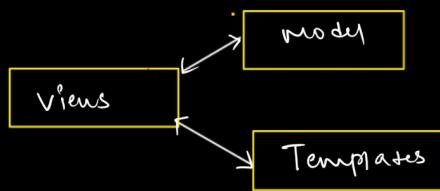
↳ Model is for the data. The data will be coming from the database & that can be linked to the 'Model' object.

2, Template is of Normal HTML. we will have HTML, CSS, frontend Javascript.

and it will also have Django Template Language (DTL).

↳ used to include Dynamic data in the page.

3, The business logics will be written in Views.



The views will use the model object & the Templates.

→ what data has to be sent on Template will be decided by View:

→ The connection to the Models will be done through View.

⇒ Model will work with data

View will work with logics

Templates is for the layout.

This is called 'Separation of Concerns',

→ we had different modules (Model, View, Template) for different concerns (data, logics, layout).

