

Django

Flask

↳ uses MVC architecture

↓
Model view Controller → to pass controls

we write
all the utilities
(or) core 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

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 sqlparse-0.2.2

Using cached sqlparse-0.4.2-py3-none-any.whl (42 kB)

Collecting tzdata

Using cached tzdata-2022.1-py2.py3-none-any.whl (339 kB)

Collecting asgiref-3.4.1

Using cached asgiref-3.5.0-py3-none-any.whl (22 kB)

Installing collected packages: tzdata, sqlparse, asgiref, django

Successfully installed asgiref-3.5.0 django-4.0.4 sqlparse-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.

(django_env) C:\WINDOWS\system32>

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

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
```

```
(env_django) C:\Windows\System32\project>
```

```
(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\krishnaleti\appdata\local\programs\python\python39\lib\site-packages (from django) (4.0.4)
```

```
Requirement already satisfied: tzdata in c:\users\krishnaleti\appdata\local\programs\python\python39\lib\site-packages (from django) (2022.1)
```

```
Requirement already satisfied: asgiref<4.0, >=3.4.1 in c:\users\krishnaleti\appdata\local\programs\python\python39\lib\site-packages (from django) (3.5.0)
```

```
Requirement already satisfied: sqlparse<0.2.2 in c:\users\krishnaleti\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>mkdir
```

```
The syntax of the command is incorrect.
```

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

```
Volume in drive D is Data
```

```
Volume Serial Number is B49E-AE1F
```

```
Directory of D:\Edu\Data Science_iNeuron\Live_Class_Notes\DJango
```

```
11-04-2022 04:11 PM <DIR> .
11-04-2022 03:45 PM <DIR> ..
11-04-2022 02:12 PM <DIR> 23. Django_iNeuron
11-04-2022 04:11 PM <DIR> django_project
```

```
0 File(s) 0 bytes
```

```
4 Dir(s) 945,152,630,784 bytes free
```

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

```
(dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\DJango>cd project
```

```
Volume in drive D is Data
```

```
Volume Serial Number is B49E-AE1F
```

```
Directory of D:\Edu\Data Science_iNeuron\Live_Class_Notes\DJango\project
```

```
11-04-2022 04:13 PM <DIR> .
11-04-2022 04:11 PM <DIR> ..
11-04-2022 04:14 PM <DIR> .idea
11-04-2022 04:11 PM <DIR> django_project
```

```
1 File(s) 692 bytes
```

```
4 Dir(s) 945,150,267,392 bytes free
```

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

This PC > Data (D:) > Edu > Data Science_iNeuron > Live_Class_Notes > DJango > django_project >

Name	Date modified	Type	Size
idea	11-04-2022 04:14 PM	File folder	
django_project	11-04-2022 04:11 PM	File folder	
manage.py	11-04-2022 04:11 PM	Python Source File	1 KB

Data (D:) > Edu > Data Science_iNeuron > Live_Class_Notes > DJango > django_project > django_project

Name	Date modified	Type	Size
__init__	11-04-2022 04:11 PM	Python Source File	0 KB
asgi	11-04-2022 04:11 PM	Python Source File	1 KB
settings	11-04-2022 04:11 PM	Python Source File	4 KB
urls	11-04-2022 04:11 PM	Python Source File	1 KB
wsgi	11-04-2022 04:11 PM	Python Source File	1 KB

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

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

```
→ (dj_env) D:\Edu\Data Science_iNeuron\Live_Class_Notes\DJango\project>python manage.py runserver
```

```
Watching for file changes with StatReloader
```

```
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
You have 11 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.
```

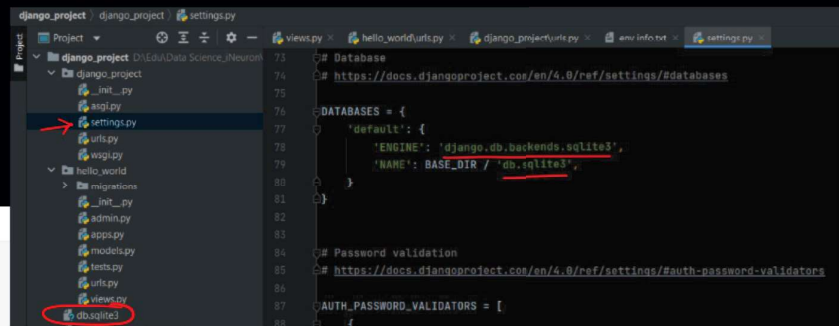
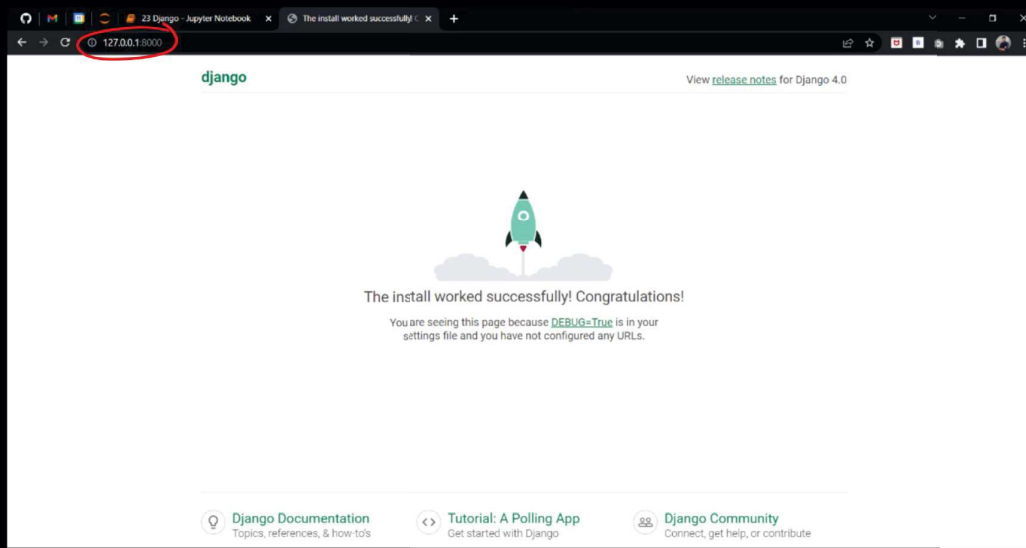
```
April 11, 2022 - 16:19:25
```

```
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.
```

Server up!



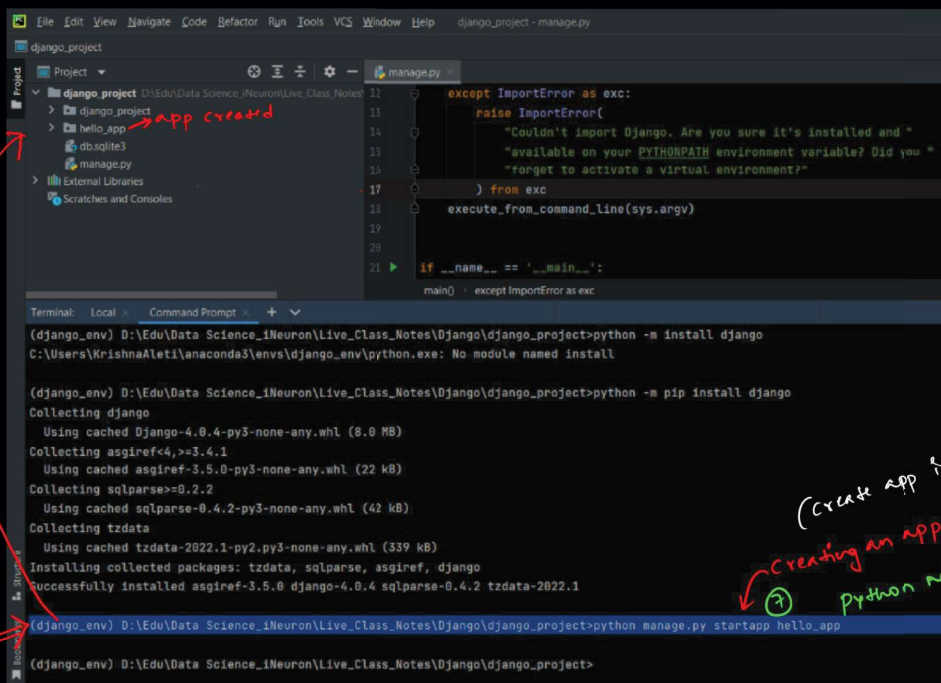
Let's look at what `startproject` created:

```
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.

When we create the project, in the folder there will be `Settings.py` & in this `Settings.py` all the major details will be present. In it by default, the database will be `sqlite3`.
⇒ `db.sqlite3` will be created in the folder.



(Create app in the `django-project` folder)
Creating an app
python manage.py startapp app-name

File Explorer path: This PC > Data (D:) > Edu > Data Science_Neuron > Live_Class_Notes > Django > django_project >

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

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5
6 def index(request):
7     text = "<h1>Hello World!</h1>"
8     return HttpResponse(text)
```

hello_app > views.py

⇒ Views.py

Contains the program logics.

⇒ 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:-

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

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

Questions

Tags

Users

Companies

COLLECTIVES

Explore Collectives

- 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.

6222

How do express

5947

How do system

5147

How ca

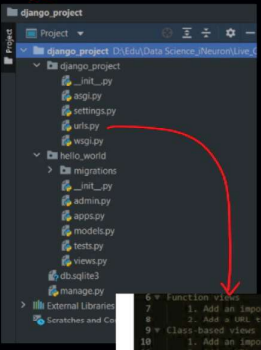
3030

Using g

4625

Accessi

Project structure:



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
django_project - manage.py

django_project
├── django_project
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   └── hello_world
│       ├── db.sqlite3
│       └── manage.py
├── External Libraries
└── Scratches and Consoles
```

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.

⇒ 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:

```

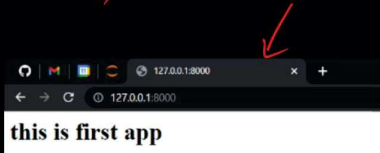
1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('blog/', include('blog.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 an other 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.

↳ Inside it another folder with project-name & `manage.py` will be created.

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.

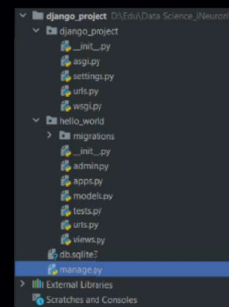
```

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

# Create your views here.

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

```

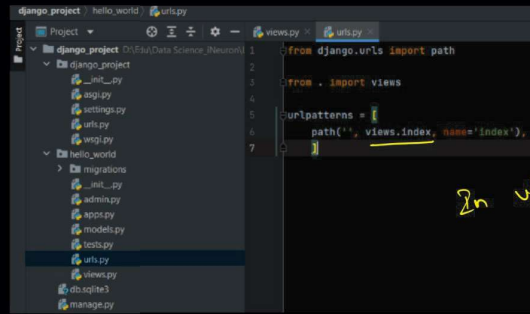


← file structure
(we always need to follow this structure only)

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

9, To Create URL config in the hello_world directory, create a file called urls.py & write the

following Code:

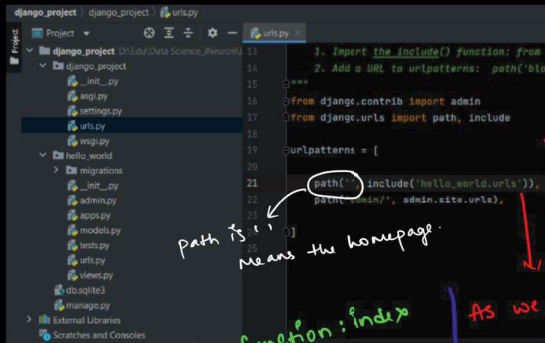


The screenshot shows the Django Project File Explorer on the left with the 'hello_world' directory selected. The 'urls.py' file is open in the editor on the right. The code in 'urls.py' is as follows:

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

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.



The screenshot shows the Django Project File Explorer on the left with the 'django_project' directory selected. The 'urls.py' file is open in the editor on the right. The code in 'urls.py' is as follows:

```
11
12
13 3. Import the include() function: from
14 2. Add a URL to urlpatterns: path('blog',
15
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20
21     path('include('hello_world.urls)'),
22     path('admin/', admin.site.urls),
23 ]
```

11, run the server

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

path is '' means the homepage.

As we had included 'hello-world.urls' → django will look into the hello-world application & its url. → In the urls file of hello-world

Flow:

⇒ write the logic/text in function: index of views.py (of app) → link it to the urls.py (of app) by giving the path

we have views.index

→ This call the index function defined in the views & the index() gets executed.

link it to urls.py (of project) by including the path as app_name.urls

⇒ when we run the server, whatever written in the views.py will be displayed.

the django will route to the paths defined in the urlpatterns of the urls.py (of the project)

→ 'app.urls' then routes to path defined in the urlpatterns of urls.py (of app) → run the server ⇒ python manage.py runserver



Reaches views.py → index() executes → 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 # Create your views here.
5
6 def index(request):
7     text = """<h1> Hey There!!! This is Krish </h1>"""
8     return HttpResponse(text)
9
10
```

function logic

we can call this 'index' in other modules as views.index

because the index() function is in views.py file.

urls.py of app

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

urls.py of project

```
views.py | hello_worldurls.py | django_projecturls.py
9 class-based views
10 1. Add an import: from other_app.views import Home
11 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('', include('hello_world.urls')),
22     path('admin/', admin.site.urls),
23 ]
24
```

⇒ 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 | hello_worldurls.py | django_projecturls.py
9 class-based views
10 1. Add an import: from other_app.views import Home
11 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('', include('hello_world.urls')),
22     path('admin/', admin.site.urls),
23 ]
24
```

routes to the paths defined

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

```
views.py
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5
6 def index(request):
7     text = """<h1> Hey There!!! This is Krish </h1>"""
8     return HttpResponse(text)
9
10
```

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 → urlpatterns = views.index

django looks into the views.py

⇒ and executes the index function Returns the result.