**6. Development of first web application**

**To start virtual environment:** myenv\Scripts\activate

python -m django startproject nurseryProject

**--------------------------------------------------------------------------------------------------------------------------------------**

**1. create project**

**2. create app**

**3. add this application in the settings.py file**

**4. create view function to provide response**

**5. define urlpattern for our view function in urls.py. So end users will use this urlpattern to send the request**

**6. runserver**

**7. send request http:/127.0.0.1:8000/hello**

**--------------------------------------------------------------------------------------------------------------------------------------**

* py manage.py

Local host IP address: [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/)

**Port number:** 8000 -----------------you can change port number manually

py manage.py runserver 7777--------------🡪 <http://127.0.0.1:7777/>

--------------------------------------------------------------------------------------------------------------------------------------

* **py manage.py startapp firstApp** ---------------------🡪 created firstApp application
* tree /f

Folder PATH listing for volume New Volume

Volume serial number is 9672-A006

D:.

│ db.sqlite3

│ manage.py

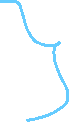
│

├───firstApp

│ │ admin.py

│ │ apps.py After creating app this files get generated

│ │ models.py



│ │ tests.py

│ │ views.py

│ │ \_\_init\_\_.py

│ │

│ └───migrations

│ \_\_init\_\_.py

│

└───myProject

│ asgi.py

│ settings.py

│ urls.py

│ wsgi.py

│ \_\_init\_\_.py

│

└───\_\_pycache\_\_

settings.cpython-311.pyc

urls.cpython-311.pyc

wsgi.cpython-311.pyc

\_\_init\_\_.cpython-311.pyc

--------------------------------------------------------------------------------------------------------------------------------------

To add this application in project use **settings.py file**

* **In settings.py file** add the name of app in INSTALLED\_APPS list.

**--------------------------------------------------------------------------------------------------------------------------------------**

**--------------------------------------------------------------------------------------------------------------------------------------**

**7.Various Practice Applications part-1**

**Views.py file: application level file**

**------------------**

Here what response should a end user get is defined in a separate function

**In views.py we** create separate functions which is responsible to provide required response.

This functions required at least one argument (request)

It takes request as a input and returns Response

For every view function we have to provide separate **urlpattern**.

By using this urlpattern end user can send request.

--------------------------------------------------------------------------------------------------------------------------------------

**urls.py project level file**

**------------**

Here we define urlpattern for our view function in urls.py file

1. Import views.py in this file
2. Add urlpattern in urlpatterns list.

Now end users will use this urlpattern to send the request

Then run server -----🡪 py manage.py runserver

Then on browser **http:/127.0.0.1:8000/hello**  (use to send request)

**How it works:**

1. First server will get request
2. Server opens urls.py file
3. Server identifies the corresponding view function from urls.py file
4. Server will execute the corresponding view function and provide required response to the end user

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**8.Various Practice Applications part-2**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**9.Templates Introduction**

In single project we can have multiple applications.

In single application we can have multiple view function.

We can reuse any application in other project.

Defining urls at project level at project level reduces reusability of the application

**Defining all application urls in a single urls.py file is always a bigger problem.**

So, maintain separate urls file at an application level.

views.py file is python file so, in this file it is highly recommended to write python code but not html code.

HTML 🡪 presentation logic

Python 🡪 business logic

So, remove html code from views.py and create a html file and write in it.

This concept is called **Templates**.

Django MVT

**M 🡪 Model (For database logic)**

**V 🡪 View (Business logic)**

**T 🡪 Template (Presentation Logic)**

Coordination --🡪 Django itself

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**10.Steps to develop template based application**

import os

BASE\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_)) 🡪 Gives directory/folder of current file

TEMPLATE\_DIR = os.path.join(BASE\_DIR,’templates’) 🡪 Creates new file in this folder

print(TEMPLATE\_DIR)

**Steps to develop template-based application:**

1. Start project.
2. Start application.
3. Add application in settings.py file.
4. **Create folder named with “templates” in main project folder. In that templates folder we created a folder named with testApp to hold testApp specific templates.**

**Ex: D:\Django Codes\templateProject\templates\testApp**

1. **Add templates folder to settings.py file so that Django can aware of our templates.**
2. **Create template html file with our required response.**
3. Create view function in views.py file.
4. Define url pattern.
5. Start server and Send request.

**11. Demo application by using templates**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**12. Template Tags**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**13. Working with static files**

How to add:

Images, css files

For that, in your main project folder

1. Create a folder: ‘static’
2. In that create image folder, css folder.

**{% load staticfiles %}** 🡪 To make all static files available to our template files(html files).

**To include css files:**

<link rel= ‘stylesheet’ href= ‘{% static “css/demo.css” %>

**To insert image:**

<img src= “{% static “images/ships.jpg” %}”>

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**14. Demo Application with static files**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**15. Working with Models and Databases Introduction**

**Model 🡪** it purely talks about Python. (Database)

It provides inbuilt database i.e. SQLite for small scale appl. to medium scale appl.

In settings.py file we have to do database configuration.

**Commands to check database configuration via terminal**

py manage.py shell

from django.db import connection

c = connection.cursor()

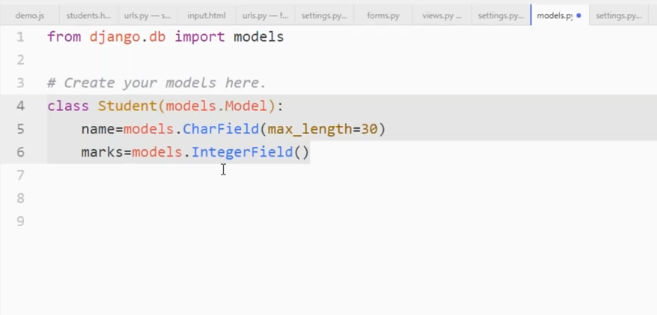
**How to configure MySQL database**

Create logical database in mysql



**How to create Table**

Create class model.py file at application level



**makemigrations 🡪** convert model classes into sql code.

i.e. database code will be generated

command: **py manage.py makemigrations**

**migrate 🡪** to create database table we have to execute generated sql code for this **migrate** command is used.

command: **py manage.py migrate**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**16. Database configurations continuation**

**Working with django-admin interface** 🡺

Adding data, viewing data …. All these things can be performed inside admin interface.

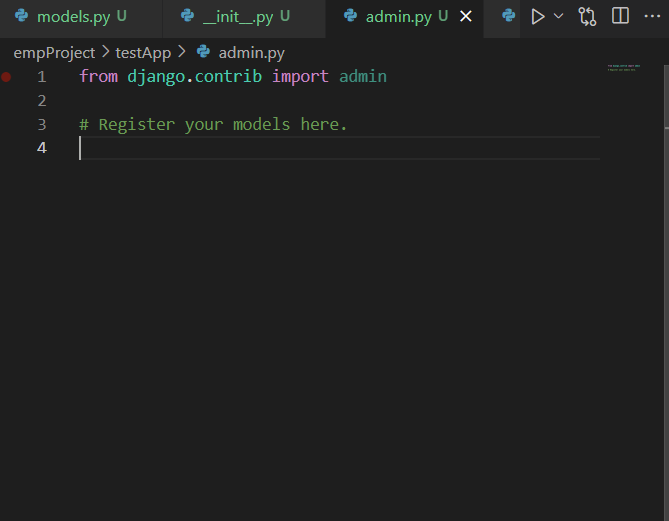
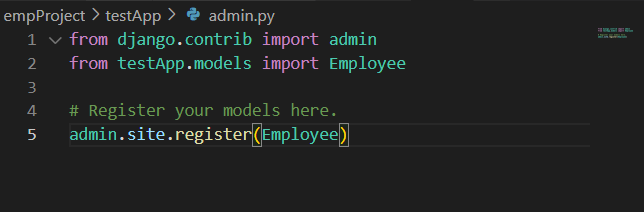
**How to login django-admin interface 🡺**

1. Create super user
2. With that you can login to <http://127.0.0.1:8000/admin>
3. But 1st you have to register for that:

Type command: py manage.py createsuperuser

Then give your username, mail id, password.

1. We have to register Model inside admin.py

We register Model class inside admin.py so that it is available inside admin interface.

But on admin interface it will not show the data

So, we have to display all columns information inside admin interface

For that you have to code in admin.py file

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**17. Demo applications with Models and Database**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**18. Demo application with Faker module to generate fake data**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**19. Working with Django forms**

**Django Forms:**

Login forms,

Registration forms,

Enquiry forms.

To grab end user’s information, we required forms.

**Process to generate Django Forms:**

1. Create a special file forms.py in application level.

form Django import forms

class className(forms.Form):

……….

1. views.py

form = forms.ClassName()

1. pass the object to template file.
2. Template file is responsible to display the forms.

{{form.as\_p}}

{% csrf\_token %} 🡪It provide security to our form.

**CSRF**: Cross Site Request Forgery

cerf\_token

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**20. Student Feedback Project with Django Forms**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**21. Django Form Validations Part-1**

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**22. Django Form Validations Part-2**

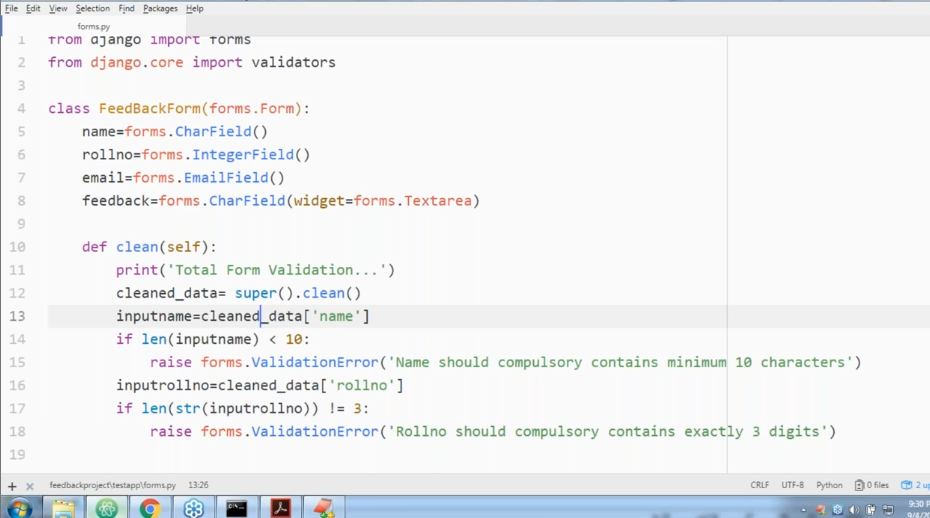
**Form Validations:**

1. Explicit Validations by using clean method.
2. Implicit Validations by using Django’s inbuilt validators.

**Implicit Validations by using Django’s inbuilt validators.**

All Django inbuilt validations are present in django.core module.

**Validating total form by using single clean() method**



----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**23. Working with Django Model Forms**

**Model Forms:**

Model Based Forms

Model 🡪 Python class which represents database table.

Create a Model:

Enter fields

Form:

model = Student

