**Django Notes**

Does not write SQL Query for Data base, table because Django itself make through its Framework.

**One Project of E-Commerce App**

in that we have Buyer, Seller and Admin

**Buyer** - Order, Wishlist, Cart, Search, Payment

**Seller -** Add Products, Discounts, Descriptions, Delete etc

**Admin -** Handle Complains, Buyer and Seller Both.

In Django does not create Admin because Django creates **automatically**

For Django have **to install Sublime Text software** Download

* **select folder** for store or save your project
* open CMD Command Prompt and select the same path open in sublime where you want to create project or save files.

**Commands for Virtual Environment :**

select the folder as per open in Sublime text

**python -m venv myvenv**

m = make , venv = Virtual Environment ,name of Virtual Environment = myvenv

**{ Eg : E:\Coding World\Python\Python\Tops\11-05-2024 - DJango Intro\Django Project>python -m venv myvenv }**

**Enter**

**Activate the Django by command :**

Sublime Path **myvenv\Scripts\activate**

**{ Eg : E:\Coding World\Python\Python\Tops\11-05-2024 - DJango Intro\Django Project>myvenv\Scripts\activate**

**}**

Install Django on that Project :

command = **pip install Django**

**{ If we want to install specific version of Django write ( pip install Django==4.0 ) }**

Django install successfully

we have to Create Project and App

Project = in that we get Admin Panel

App = In thar we create User panel

Create App

**(myvenv) E:\Coding World\Python\Python\Tops\11-05-2024 - DJango Intro\Django Project>django-admin startproject mysite .**

using command { **django-admin startproject mysite .** }

mysite = it is name of file it would be anything

( . ) = Dot represent the Current Folder

After Successfully install

**change Time Zone in setting.py in mysite**

**UTC change into = Asia/Kolkata**

Manage and Runserver :

**(myvenv) E:\Coding World\Python\Python\Tops\11-05-2024 - DJango Intro\Django Project>python manage.py runserver**

In your Browser check status your file or project install successfully or not

command = **localhost:8000**

**it is all setup of User Panel Home Page**

**For Admin**

**All this for User Panel**

After virtual Environment > Activate the same > pip install Django > Django-admin startproject mysite . > python manage.py runserver

**Check the Admin Panel Status in browser :**

**localhost:8000/admin**

**username = akshay8281**

**Password = Test@123**

* **In admin it creates tables and all automatically by below commands**

**python manage.py migrate**

* After successfully creates table than run the server

**python manage.py runserver**

* to stop the server command are

**CTR + C = to stop the server in CMD (Command Prompt)**

* TO set the username in Admin Panel

**python manage.py createsuperuser**

(When set username make sure it is in small letters)

* **After username for admin ask for Email but it is optional**
* After Email ask for password please fill simple password at initially createdceg. test@123  
  **{ Password is not visible in CMD Prompt }**

Run the Server to check whether the command is work or not. **{ python manage.py runserver }**

After you add username and password successfully than go for check whether the server run properly or not.

**For User Panel = localhost:8000**

**For Admin Panel = localhost:8000/admin**

**python manage.py runserver**

**PROJECT = ADMIN PANEL**

**APP = USER PANEL**

* After that we have to create App for our project
* Stop server if run and go into CMD Prompt

Command to create App for the project

**python manage.py startapp myapp**

**{ Path for the Project> python manage.py startapp myapp**

**}**

Startapp = create app command

Myapp = it is app name for our project and it would be name anything

Project = mysite

Mysite = Project

App = myapp

We have to registered our app in **mysite(project) setting.py** in that **INSTALLED\_APP** we have to put our project name which is **myapp**

**‘myapp’,**

We have registered our app as myapp successfully.

**Django Interview Question**

The Django framework follows which architecture?

**MVT = Models Views Templetes**

Now we discuss about files in myapp as per **MVT (Models Views Templetes)**

* **In models.py we have to put data of CLASS. Also we create CLASS as we want as many in that file only.**
* **Whatever Tables are created have through CLASS and put it in model,py only**
* **In Views.py we created in that are All functions for the requirement in Project.**
* **Views.py** is the **main business logic** in any **project. Main coding is written in Views.py.**
* The **templates** does not show in any folder of **Project** ,so we have to **create manually folder** in the **myapp.**
* The **main view part** (display in screen using **HTML** and **CSS**) will be **shown in templates folder.**
* **All the request will sent on views.py**

When we **create file index.html** in **templates** **folder** and in that some data on it, after that the process flow for that shows in **localhost:8000** steps below :

1. When server firstly goes in our project, in our project goes into urls.py.
2. In that shows only admin panel, so we include our page that we have import library by the few commands and few steps in **urls.py**  
   **from Django.urls import path,include**
3. We have **create** our **root path (localhost:8000 or our Home Page) in mysite urls.py  
     
   path(‘’,include(‘myapp.urls’))**
4. After that make **new urls.py in myapp** and copy paste the content **from mysite urls.py** as below

**from django.urls import path**

**from . import views ////** All Functions are imported from views

**urlpatterns = [**

**path(‘’,views.index,name = ‘index’), ///** name of the function in views

**]**

In **views.py** function is to create **index** but the **parameters** of all the functions would be **common is request and function return render the html file by request**

def index(request):

return render(request,'index.html')

After that start the app by below command :

**myvenv\Scripts\activate**

**python manage.py runservr**

**SHORTCUT STEPS FLOW OF HTML SHOWS IN DISPLAY**

**Localhost:8000**

**> Project(mysite) urls.py /// Check root of myapp.urls.py**

**> myapp(urls.py /// check the root in myapp urls.py) and goes into views.py**

**> views.py (/// check the functions mention in myapp.urls.py and render request of html file)**

**> templates (index.html)**

**Successfully page load in Display localhost:8000 then**

* Import bootstrap by cdn links
* Navbar imported from bootstrap
* Navbar is common for all the pages of the website or App. So create header.html for that and remove all header condent from other files.
* So we have connect header file to other HTML files,we can use use jinjako Method

Example

In **index.html** file **import from header.html content** :

**{% extends "header.html" %}**

**In header.html**

**{% block content %}**

/// block = main syntex

/// content = name mention anything you want

/// Start the Content where you want to display the from html file.

**{% endblock %}**

/// endblock = main syntex

/// End the content of that particular html file where you display to the screen

**/// Connect Page**

In Django, html files does not link directly via ancor tag or links. Does not allowed by Django to link directly.It Follows MVT Method = Models Views Templates. Using urls.py .

All request goes in urls and then goes into views

**It is used Jinjako**

**header.html**

In ancor tag **{% url ‘contact’ %}** /// make it small letters

**Then goes into myapp > urls.py**

New Path have to be mention in urls.py

**path('contact/',views.contact,name='contact')**

**After that goes into Views.py**

Have to create **new Request** for **contacts** as per below

**def contact(request):**

**return render(request,'contact.html')**

**Same as for others**

After Form is created in contact.html then mention the name,method and action in form

**name = contact**

**method = “post”**

**action = {% url ‘contact’ %}** /// Goes into urls.py > views.py > contact.html

Note :

* Suppose we have two url of contact name in **header.html** as well as in **contact.html.The should be same.**
* So confusion is that which **contact url** is to be **CALL** to run **header.html** **or contact.html**
* In **header.html** we created **form** and **contact name** is **mention in ancor Tag** so by **default** it is **call GET METHOD**
* And in **contact.html** we already **mention METHOD = “POST”.** When there is **no any method** mention by **default call GET METHOD** .**Both have different Method header.html = GET METHOD and contact.html = POST Method.**
* So for that we can use **IF ELSE** in **views.py** for **contact**
* We have to **create CLASS in models.py** for the **contact** to **submit the Data**.
* pass means without error with empty function

**In Models.py**

**from django.db import models**

* **models** is imported **library** from the python
* **models** = **is** **modules** - Readyment created file by python
* **In that models have python own class = model**

**class Contact(models.Model):**

* **Django Framework most Advantages by default gives field ID which is Primary key.**
* **We created four field but Django gives 5 fields including ID Field**
* **When we create CLASS in models.py after that we have to successfully create a table below are the commands**

**Open CMD Command Prompt**

* **python manage.py makemigrations ///** This is only script to execute the class created in models.py
* **python manage.py migrate ///** After that Execute the migrations file from the CLASS of models.py here are the command
* **python manage.py runserver** /// After that run the server and check that correctly work or not
* Table is created successfully
* After **Table created** then first we have **to registered our models to admin.py**
* After that in admin.py we have to import **Contact** **class** from **models.py**  
  **from .models import Contact**
* We have to register in admin through below syntax :  
  **admin.site.register(contact)**

**After that goes into view.py :**

Import the Contact model

**From .models import Contact // Contact Class imported from models.py**

**ORM - Object Relational Mapping**

**First time we are writing ORM query in views.py**

def contact(request):

if request.method=="POST":

**Contact.objects.craete(**

**name = request.POST['name'],**

**email = request.POST['email'],**

**mobile = request.POST['mobile'],**

**remarks = request.POST['remarks'],**

**)**

else :

return render(request,'contact.html')

* After that whenever **we work on form in Django using POST METHOD compulsory we have to add in JINJAKO {% csrf\_token %}** in the form in contact.html.
* It is put just after the Opening Form Tag.
* **(% csrf\_token %)** = It is security token to work with the Data comes from method of POST does not access from unauthorized url.It makes secured your Data.
* When we to **transfer** a **msg** or **Data** from **view.py to HTML** file we have to **compulsory put into dictionary format in return**.  
  **return render(request,'contact.html',{'msg':msg})**
* In **contact.html** put **MSG** after the contact header and apply Jinajoko for the msg when for submitted **{% if msg %}**

**<b style="color: navy;">{{msg}}</b>**

**{% endif %}**

* In Admin Panel, the saved data does not shows the name of people,it shows object of the user. So that reason go into models.py and make function for the same **def \_\_str\_\_(self):**

**return self.name**

**Date 18-05-2024**

Work on Reviews :

* Create a variable **contacts** put below in syntax in **view.py**
* **contacts = Contact.objects.all()** //all will give data in list form
* data put into return as per **msg**
* copy data of if in return and paste it in else: and then remove the msg in that
* print the data in html of contact.html put data using jinjako   
  <h1>Recent Users</h1>

<hr>

{{contacts}}

**Sign Up**

* Copy the data of CONTACT.html
* When fill up the form we have to **store data into the DATABASE** through Table.
* **We have to create CLASS in models.py for signup or User**
* Store all the data which want from user except Confirm password.
* **After CLASS created** then run the command in CMD   
  **python manage.py makemigrations**
* And **after successfully migrations** then run below command  
  **python manage.py migrate**
* Migrate successfully the run the server  
  **python manage.py runserver**
* After successfully migrations and migrate then we have to register that class in admin.py
* And then goes into views.py
* When Signup by the user we have to check **that email is registered or not previously**
* We have to also **check** the **password and confirm password is Match or not in server side**. It is also check in client side or frontend side validation for the password.
* How to check email ID registered or not in server side or database  
  we can use try : in that have a method  
  **User.objects.get()** ///get return single object from User
* First check **email id is registered or not** using **try get method** if **already** registered then goes into **signup** mage and shows **msg** **already** registered email id
* In signup up we have to check

**LOGIN**

* Copy the **signup.html** to the **login.html**
* In **login.html** data **insert** from **urls** will be **goes into views.py**
* We have to **check the email id is registered or not while login by using try and except method in that also get() method.**
* After **email check** then we have to **check password is correct as per email id mention database of user or not.**
* When l**ogin email** id and **password** will **match** **before that we have to create session for that particular user**. **Session** **means** a **particular user data**
* In session, with the help of request and use utlity called session.  
  in session we have to create variable **. Created for email because each have unique email id**
* When **login successfully** after return index.html then we have **make logout button in the place of login when session or user logged in.**
* In **header.html** use **Jinjako if else** in that login and signup button is shows when login or **session is not get**. While **session will get** then **shows logout button** and we also **make url for logout button**.
* Create logout in urls and in that delete all the session which is created and shows msg when deleted the session

**21-05-2024**

**Change Password**

* In header.html have to add button for Change Password and also make url for the same.
* In urls.py create the path for the change password and make sure in views.py does not create using – (- Hyphen Sign).Make it in \_ (\_ UnderScore) for views.py only.
* Create new file to change-password.html
* Change password is only possible when user is logged in.
* So we have to change password first of all we have to fetch the data of password from database of that particular user through SESSION already created in login.html.All this code in views.py
* There many steps in Videos kindly refer video 21/05/2024

**Forgot Password**

* First Check the Email ID is registered or not in our database or not.
* If email is registered already then send OTP for that user only
* Create new Page for OTP for user put the otp they got.
* And then check OTP inserted is correct or not with Generated OTP to the user in their Email ID.
* After enter correct OTP then create another Page for New Password
* And then check the New Password and Confirm New Password and that would be updated in session of email id.
* If email id entered and it is not registered then shows error “Email ID is not Registered”
* When Email is correct > OTP insert is incorrect then throws an Error “OTP is incorrect” and Try to insert same OTP
* When email is correct > OTP is Correct > New Psw – is Incorrect with New Psw than throws error “New Password and Confirm New Password does not Matched”

Steps

* First add the Jinjako in login.html for forgot Password
* Make path for forgot password in urls.py
* Go to views.py and create forgot\_password function
* Create forgot-password.html and copy the data of login.html and changes the same according to the requirement of Forgot Password
* Create path in urls.py
* After make function views.py get the data through POST request and get email id status registered or not.
* If not then throws an error “Email not registerer”.
* Data of email is check from the session of email we crated earlier
* OTP will send on gmail id so we have to search services for the Gmail.
* So we depend on Gmail API intigrations.
* Search on google **send Gmail Django** ( **https://www.geeksforgeeks.org/setup-sending-email-in-django-project/** )
* Put below syntax into settings.py in the last  
  **EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'**

**EMAIL\_HOST = 'smtp.gmail.com'**

**EMAIL\_USE\_TLS = True**

**EMAIL\_PORT = 587**

**EMAIL\_HOST\_USER = #sender's email-id**

**EMAIL\_HOST\_PASSWORD = #password associated with above email-id**we cannot change above 4 lines in hi syntax

**Step - 1**

* **In EMAIL\_HOST\_USER =** [**aks.pitroda.8@gmail.com**](mailto:aks.pitroda.8@gmail.com) **//mail will be send from owner**
* **EMAIL\_HOST\_PASSWORD =   
  /**/ we cannot store our original password in this section so we use gmail
* So we login **our (owner)** **email** id into chrome and go into **APP PASSWORD** and then **Give Project Name** where we use and then **code is generated** **copy** that and **paste** into our projects > **settings.py > EMAIL\_HOST\_PASSWORD = ‘digu ejjq amvd egvn’**

**Step – 2**

* We have to send mail so we import following things into views.py **from django.conf import settings**

**rom django.core.mail import send\_mail**

* Also import random to generate random number.

**Step – 3**

* Copy the API of Django from gmail services
* Paste it into Forgot Password Function

**subject = 'welcome to GFG world'**

**message = f'Hi {user.username}, thank you for registering in geeksforgeeks.'**

**email\_from = settings.EMAIL\_HOST\_USER**

**ecipient\_list = [user.email, ]**

**send\_mail( subject, message, email\_from, recipient\_list )**

* After changes in above syntax and return it into **otp.html**
* Before that we have to **create session for the email\_otp and OTP to access globally.**
* Create otp.html and copy paste the data of forgot password.html and changes the same according to otp.
* Go to urls.py to create path for the verify-otp
* And then go to views.py create function verify\_otp
* In that we have to create new file name new-password.html after successfully entered OTP.
* Copy change-password.html to new-password.html after OTP input
* New url created is new-password and put that into urls.py
* After that create **new function** **new\_password** in views.py

**21-05-2024 (Part 2)**

**Profile Image Set**

* For Profile picture have also **Third Party API** to set the profile picture.
* Every user have own Profile Picture
* This field is created after the **session** created **makemigrations** and **migrate** and after the **table** is created.
* So entry will put in the **models.py**. Add new field in   
  **class User**  
  **profile\_picture = models.ImageField(upload\_to = “profile\_picture/”)**/// Compulsory Attribute is upload\_to

**Step - 1**

* Search for **API** import links by **upload image Django** in Chrome.
* Put below data from **upload image Django** in **settings.py  
  MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')**

**MEDIA\_URL = '/media/'**

* Also **import os in settings.py** and put it in top of the file
* **MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')** we import the os then **BASE\_DIR** creates files or folder name called media.
* **/ means = root  
  goes into media = /media  
  and then our url will be created**

**Step - 2**

* We have to below links in our project urls.py = mysite > urls.py  
  **from django.conf import settings**

**from django.conf.urls.static import static**

* And then after below **condition** will **put last** of the code in **mysite urls.py  
  if settings.DEBUG:**

**urlpatterns += static(settings.MEDIA\_URL,**

**document\_root=settings.MEDIA\_ROOT)**

* And then we have to install library for Profile Picture in CMD **Prompt  
  Command :  
  pip install pillow**
* After successfully installed pillow then give command  
  **python manage.py makemigrations**
* After makemigrations then give command:  
  **python manage.py migrate**
* After that check the status working or not in Admin panel
* When user is sign up in the app give option to upload image.
* When we have input type = “text” and input type = “file” so we have to put attribute compulsory in the form Tag  
  **enctype = “multipart/form-data”**
* Then data will goes into views.py and put the data in signup function  
  **profile\_picture = request.FILES['profile\_picture'],**
* **Profile Picture** will shows in **header.html** show we have to use **session** to show the profile picture.  
  session is created in **login function in views.py   
  request.session['profile\_picture'] = user.profile\_picture.url**
* **Delete the session after user logout ,data put in logout function  
  del request.session['profile\_picture']**
* **After that go into header.html and put the session in image tag of src  
  <img src="{{request.session.profile\_picture}}" style="width: 50px; height: 50px;">**

**23-05-2024**

**Profile Update**

* When click on Profile photo profile should be updated
* Go to header and make profile img tag into ancor tag
* Make url profile in that ancor tag of profile img in header.html
* Make url path in urls.py and then after create function of profile in views.py and create profile.html.
* Copy the data of signup.html into profile.html
* Make changes according to the requirement
* When click on profile picture shows the data of that particular user
* Fetch the user objects