DJANGO

⇒ Django is a Python framework to develop web applications.

⇒ The Django is used to develop backend side of a website

⇒ The languages like html, CSS, JS are used for front-end development.

* Other languages uses MVC
  Model - View - Controller

⇒ But, Django uses **MVT**
  Model - View - Template

⇒ To check the version of the installed django ⇒ django-admin --version.

⇒ Install virtual envcovapper to not to disturb the version of the django

* pip install virtualenvwrapper-win.

⇒ To create an environment Syntax.

mkvirtualenv (your optional name)
ex

* mkvirtualenv test

⇒ To Create a folder in virtualenv

* mkdir Projects.

go into this folder by using cd projects.

⇒ To start a Project - syntax.

* django-admin startproject telusko.

⇒ To open the server

* Python manage.py runserver.

IMP * To make dynamic Content in webpages we use django.

⇒ To Open a created virtualenv

* workon (created name)

* workon test

V.V.IMP ⇒ To Create apps in your Project.

* Python manage.py startapp (name)

Ex: * calc

⇒ Creating a "Hello World" text

* Create 'urls.py' in the app.

Syntax:

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

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

for mapping

⇒ In views.py of app create home function

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello world")
```

⇒ Modify the main urls.py file

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

urlpattern = [
    path('', include('Calc.urls')),
    path('admin/', admin.site.urls),
]
```

url of app.

Templetes

⇒ The basic idea is that instead of writing/creating html tags in views.py we can make seperate folder for the html content. (tags)

⇒ Create a directory in your current Project folder
ex templetes

⇒ In this you have to create a html file
ex home.html

⇒ Go to settings.py, modify Templates

'DIRS': [os.path.join(BASE_DIR, 'templates')]
↓
name of folder

⇒ change views.py in app.

def home(request):
return render(request, 'home.html')
↓
file name

(or)

The name to be dynamic.

```
def home(request):
    return render(request, 'home.html',
                  {'name': 'cherry'})
```

* Also, change the syntax in html page

```
<h1> Hello {{ name }} !! </h1>
```

Part 2 Templates

To create basic code for all the pages

* Create another html file in templates folder

```
<body bgcolor = 'cyan'>
    {% block content %}
    {% endblock %}
</body>
```

→ above header

=> In the first html file

```
{% extends 'base.html' %}
    {% block content %}
    <h1> Hello {{ name }} !! </h1>
    {% endblock %}
```

Adding two numbers

⇒ Modify the home.html in block content, Create forms tag.

```
<form action="add">
    Enter 1st number: <input type="text" name="num1"><br>
    Enter 2nd number: <input type="text" name="num2"><br>
    <input type="Submit">
</form>
```

⇒ Create result.html

```
{% extends 'base.html' %}
{% block Content %}
Result... : {{name}}
{% endblock %}
```

⇒ Create add function in views.py

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

⇒ add a path in urls.py of app.

Path('add', views.add, name='add')

GET vs POST

Get → used to fetch data by using this method our passwords are seen in the address bar

Post → Post is used to submit the data to the server and it do not shows any password hints in the address bar.

⇒ so in the home.html use syntax.

```
<form action='add' method='Post'>
{% csrf_token %}
Enter 1st num: <input type="text" name="num"><br>
Enter 2nd num: "  "
<input type="submit">
</form>
```

Also change the Get method to Post in views.py

Put :- It is used to update the data.

Delete :- It is used to delete the data.

Model View Template

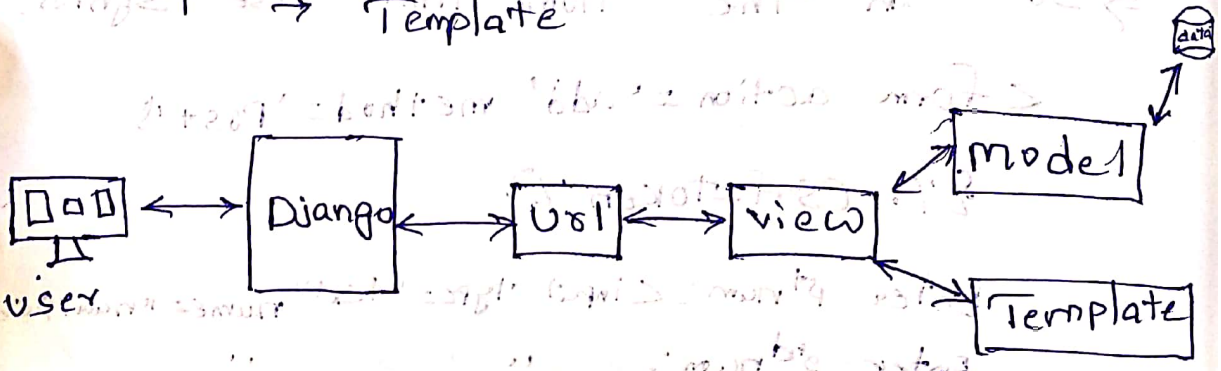
There are 3 things in a website

1) Layout
2) Data
3) Logic

* Logic is written in views file.

Business Logic

M → Model ↔ data
V → View
T → Template



MVT Framework

Static Files

Download → Travello, Colorlib

A website

⇒ Create a new app "Travello" in Telusko directory

⇒ Copy the index file of Travello to the templates dictionary

⇒ Create urls.py in travello.

```
from django.urls import path

from . import views

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

⇒ Modify the views.py of travello.

```
from django.shortcut import render

def index(request):
    return render(request, "index.html")
```

⇒ Modify the urls.py of main directory (Project Telusko)

```
from django.urls import path, include

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

⇒ Runserver

If you inspect this page, there are a lot of errors

⇒ Create a folder in the main project telusko to add (copy) the remaining files of the website (css, JS files).

Ex static

⇒ Copy all the folders of the website to static folder

⇒ we have to set a path so, the django can able to add the remaining files to integrate website.

* In settings.py of Telusko

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]
STATIC_ROOT = os.path.join(BASE_DIR, 'assets')
```

↓ An optional name

⇒ we have to create a folder for the mentioned assets

SYNTAX :-

```
Python manage.py collectstatic
```

⇒ Edit index.html file

add to the top in file

```
{% load static %}
```

After that, you have to edit the Path of the stylesheets in html

ex:- <... href="{% static 'styles/bootstrap4.css' %}">

You have to call (or) tell the django to locate the stylesheets in the perticular folder called static.

→ Runserver you notice the changed.

modify the code in the file where you found the path elements like href, src etc.,

* Just integrate {% static .... %}

Runserver you will get the website

Passing Dynamic Data in Html

* In the above website all the data is static, if you want to make it dynamic --

* The layout is fixed but you can change the data in it.

ex:- if you want to change the Price of any perticular bicity.

→ In views.py.

```
def index(request):
    return render(request, 'index.html', {'Price': 700})
```

Also you have to change the syntax in the index.html file, for which city you want to change

ex:- Change the price of Bali

Syntax

```
<div class="destination-Price">from ${{Price}}</div>
```

⇒ IF you want to change all the stuff like name, description, price, image

* we can create objects - class which will hold all the (4) data

→ models.py in the django is for database purpose.

we can also create classes in this file.

Note variable id is for database purpose of future.

SYNTAX:

```
class Destination:
    id : int
    name : str
    img : str
    desc : str
    price : int
```

⇒ we have to Create Objects in the views.py file

```
from .models import Destination

def index(request):
    dest1 = Destination()
    dest1.name = 'Mumbai'
    dest1.desc = 'The city that never sleeps'
    dest1.price = 700
    return render(request, 'index.html', {'dest1': dest1})
```

Also you have to change the index.html

for name

```
<div class="destination_title"><a href="destinati.html">{{ dest1.name }}</div>
```

for desc

```
<P> {{ dest1.desc }} </P>
```

for Price

```
<class="destination_Price"> From {{ dest1.price }}</div>
```

☹ Do this for the 3 Places/Citys in the website.

And delete the remaining city files in index.html file.

⇒ Just Place one destination in html and we can insert it into a loop and run for our desired times.

for loop in html

Modify views.py

```
dest1 = Destination()
dest1.name = 'Mumbai'
dest1.desc = 'The City That Never Sleeps'
dest1.img = 'destination_1.jpg'
dest1.Price = 700

dest2 = Destination()
dest2.name = 'Hyderabad'
dest2.desc = 'First Biryani, Then Sherwani'
dest2.img = 'destination_2.jpg'
dest2.Price = 650

dest3 = Destination()
dest3.name = 'Bengaluru'
dest3.desc = 'Beautiful city'
dest3.img = 'destination_3.jpg'
dest3.Price = 750
```

* lets pass it as one object ⇒ `dests = [dest1, dest2, dest3]`

```
return render(request, 'index.html', {'dests': dests})
```

* The images are in the assets folder
loop on html

* One class is enough in models.py and we are created many objects in views.py

⇒ In index.html ↓

Dont use zinza inside zinza.

{% load static %}

{% static "images" as baseurl %} → for img tags

In destination tags:

{% for dest in dests %}

< img src="{{ baseurl }}/{{ dest.img }}" alt="">

{{ dest.name }}

{{ dest.desc }}

{{ dest.Price }}

{% endfor %}

Using -if- statement

* In website special offer is seen on all citys to modify that ↓

modify the class in models.py add

```
Class Destination:
    id: int
    .
    .
    offer: bool
```

Also modify views.py

```
dest1 = Destination()
dest1.offer = False

dest2 = Destination()
dest2.offer = True

dest3. Destination()
dest3.offer = False.
```

we have to change the code in index.html

```
{% if dest.offer %}

"<class= spec-offer text_center">.....

{% endif %}
```

Now, You Can see the offer only on the Hyderabad. 🙂

⇒ In Django we Can Connect to database with the help of Postgresql

Download Postgres

You can also download Pgadmin to interact with the postgres.

Object Relational Mapping (ORM)

→ ORM is a Concept where you Can Create tables in database with the help of classes in your application.

→ The objects in our classes are the data in the tables.

→ You are not supposed To write any SQL Queries.

Models and Migrations

→ Create a database 'telusko' in pgadmin

→ To Connect with Postgresql server you have to modify settings.py of telusko.

```
DATABASES = {
    'default' : {
        'ENGINE' : 'django.db.backends.postgresql',
        'NAME' : 'telusko',
        'USER' : 'Postgres',
        'PASSWORD' : 'Cherry@143',
        'HOST' : 'localhost'
    }
}
```

← You can also give IP adress of machine (cap)

⇒ we want a Connector to Connect the Postgres with Python.

Use: PSYCOPG2

SYNTAX: PIP install PSYCOPG2

⇒ The table which you Create is depend upon your model in Python.

for that we have to Convert a class into a model.

In models.py

→ For images you have to specify the Path of the image location.

* refer Model fields in google django

```
class Destination(models.Model):
    name = models.CharField(max_length=100)
    img = models.ImageField(upload_to='pics')
    desc = models.TextField()
    Price = models.IntegerField()
    Offer = models.BooleanField(default=False)
```

* If you want to create a table you have to migrate your data for that Command

Syntax

Python manage.py makemigrations

error = ?

Go to Settings.py

Add 'travello.apps.TravelloConfig' in the INSTALLED APPS

* Also install Pillow package

☺ A folder is created in migration folder

→ To check the Code

Python manage.py sqlmigrate travello 0001

→ To make actual migrations

Python manage.py migrate

* You Can check the Tables in telusko database in Pgadmin ☺

→ If you are forgot any Code in models modify it then remigrate it

Python manage.py makemigrations

option : 1

exit : 0

Python manage.py migrate

Admin Panel

→ The data is pushed into the server by 2 ways

1. User.
2. An admin page.

→ ☺ Django has its own admin page

localhost:8080/admin

It shows you

* To Create a Super user in Terminal

⇒ Python manage.py help

You will get list of Commands

⇒ Python manage.py createsuperuser

Username : Charan

Email address : akala...@

Password : 1234

[Y/N] : Y

Created ☺ superuser.

* Login in admin page

⇒ We can also add destinations in our travello website by using this admin page

⇒ In admin.py of travello

You have to register the model.

syntax:-

```
from .models import Destination

admin.site.register(Destination)
```

* Go back to browser you will get the Destinations tab, which you can enter the Credentials of the city. 😊

→ 😞 The Problem is while uploading images Initially they are static, now we have to make them dynamic.

Go to settings.py

* Add the path for media.

At last STATIC_URL ↓

```
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

⇒ we have to also change urls.py of Travello

```
from django.conf import settings
from django.conf.urls.static import static

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

urlpatterns = urlpatterns + static(settings.MEDIA_URL,
    document_root = settings.MEDIA_ROOT)
```

* Just google all the command. 😊

⇒ Add data into the website by admin page, destinations

⇒ All the data is updated into the server but images Syntax must be changed in 'index.html'

⇒ Also change views.py remove the static objects

```
def index(request):
    dests = Destination.objects.all()
    return render(request, "index.html", {'dests': dests})
```

* Inspect the page if any errors..?

⇒ change the img tag of destinations

<img src="{{ dest.img.url }}" alt="">

⇒ Your new Dynamic website is ready 😊.

User Registration

1. You can add users through admin page

(or)

2. You can make your webpage to accept the user registration. By creating form.

⇒ we have to create seperate modules for that so that you can use in many other apps.

Python manage.py startapp accounts

* Create urls.py in accounts

```
urlpatterns = [
    path("register", views.register, name="register")
]
```

* Also add url of accounts to main urls.py

in urlpatterns

```
path('accounts/', include('accounts.urls'))
```

* In views.py

* Modify index.html

in the header remove "services, news tags, add

~~
register
~~

register

⇒ Create register.html page in templates

<title> Registration </title>

<body>

<form action="register" method="post">

{% csrf_token %}

<input type="text" name="first_name" placeholder = "First name">

also create for lastname, email, password1, 2

<input type="submit">

</form>

</body>

* You don't need to bother that a table is already Created for the users in the pg-admin server.

ex:- auth_user.

* Run the Server the page is ready. ☺

⇒ If you want to save the user details in the database

* In views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth

def register(request):
    if request.method == 'POST':
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        username = request.POST['username']
        password1 = request.POST['password1']
        password2 = request.POST['password2']
        email = request.POST['email']

        if password1 == password2
            if User.objects.filter(username=username).exists():
                print('username taken')

            elif user.objects.filter(email=email).exists():
                Print('email taken')

            else:
                user = User.objects.create_user(username=username,
password=password1, email = email, firstname = first_name,
lastname = last_name)
                user.save();
```

print('user created')

else:

print('password not matching')

return redirect('/')

Passing messages to user

⇒ we are printing the error messages in the Console.

⇒ we need to print them in the page itself.

for that we have to modify the views.py

```
from django.contrib import messages
```

In the place of print

```
messages.info(request, 'Username taken')
```

email - taken

Password not matched

```
return redirect('register')
```

* Also modify the register.html Page after <forms>

```
{% for message in messages %}
    <h3> {{message}} </h3>
{% endfor %}
```

User login

Create a page login.html

```
<form action="login" method="post" >
{% csrf_token %}
<input type="text" name="username" placeholder="username"><br>
<input type="password" name="password" placeholder="password"><br>
<input type="Submit">
</form>
<div>
    {% for message in messages %}
    <h3> {{message}} </h3>
    {% endfor %}
</div>
```

* Also add the Path to urls.py

```
Path("login", views.login, name="login")
```

Modifg the views.py

```
def login(request):
    if request.method == 'POST':
        username = request.POST['username']
        Password = request.POST['Password']
        user = auth.authenticate(username = username,
                                 Password = Password)
        if user is not None:
            auth.login(request, user)
            return redirect('/')
        else:
            messages.info(request, 'invalid Credentials')
    else:
        return render(request, 'login.html')
```

* Create a login in index.html

```
<li><a href="accounts/login">Login</a></li>
```

User logout

* In home page if the user is loged in you have to show the logout option and his name.
  by avoiding register, login options.

Modify index.html

```
{% if user.is_authenticated %}
<li> Hello, {{ user.first_name }} </li>
<li> <a href="accounts/logout"> Logout </a> </li>
{% else %}
<li> <a href="accounts/register"> Register </a> </li>
<li> <a href="accounts/login"> Login </a> </li>
{% endif %}
```

* Add url in urls.py

```
Path ("logout", views.logout, name="logout")
```

* Also Create a method in views.py

```
def logout (request):
    auth.logout (request)
    return redirect ('/')
```