Django

Software

- software is nothing but set of instruction or collection of program's which are used for either performing a specific task or operating a system.

Program

Program is a set of instruction's.

Instruction

set of rules and regulation's that will make computer to do a task.

Classification of software

Software are classified into two types.

System Software Application Software

System Software

Software which responsible operating and controlling a system. is known as system software.

OR

Software which responsible creating a platform where we can run the other software.

Example-

Operating System (OS) like Windows, Linux, Mac Os etc.

Application Software

Software which responsible for performing a specific task.

Classification of Application software

Application software classified into three type's.

Stand-alone software Web Application software client-server software.

Stand-alone Software

Software which is not dependent on either network connection or operating system.

Example ms-office, ppt, calculator, Ludo king etc.

Web Application Software

Software which accessed through web browser.

Example Bet 365

client-server Software

client-server software is a distributed application structure which are used for dividing the task between providers (server) and service requesters (client's).

Example World Wide Web (WWW).

Software Architecture

Software architecture is a designed pattern. Where we can create software.

classification of software architecture

One-tier Architecture Two-tier Architecture

N-tier Architecture

One-tier Architecture

- In one-tier architecture there is only one phase.
- In one-tier architecture we will use only Front-end technology's like HTML & CSS for designing the application.
- All the modification in the application done by HTML page.
- We can create only static application by using one-tier architecture.

Example - School website, small scale

Two-tier Architecture

- In two-tier architecture we have two phase.

1- Front-End

2- Back-End

- We are going two use Front-End technology for displaying the content. We are going to use python programming language to make some changes in back-end.
- Semi-Dynamic webpage's are created by using two-tier architecture.

Example Industries.

N-tier Architecture

- In n-tier architecture we have three phase.

  1- Front - End

  2- Back - End

  3- Data Base

- We can create Dynamic application using N-tier architecture.

  Example FaceBook, Instagram

Frame - Work

- Frame-Work is used for providing a prototype.
- Fream-Work is a software which is designed for supporting the development of the application.
- The main goal of the frame-work is to allow developers to focus on components of the application that are new instead of spending time on already developed components.

Web - Frame Work

Web frame-work are used for developing web application.

Classification of Frame-work

Frame-Work are classified into two types.

1- Major Frame-Work

2- Micro Frame-Work.

Major Frame-Work

- By using major frame-work we can create an application which are suitable for large scale industries.
- Many people can access the application at same time.

Example Youtube, Pubg, FaceBook etc...

Micro Frame-Work

- By using micro frame-work we can create application which is suitable for small scale industries like school & collage's.

* Draw Back of Micro frame-Work -

Application can't be accessed by many people at a time

Example school & collage websites.

Frame-Work of different programming / scripting language.

JAVA - Spring Boot, Hybernate, structs

C# - .Net

PHP - Laravel, Phalcon

PYTHON - Major Frame-Work -> Django, Pyramid

Micro Frame-Work -> Flask

App's designed by using python

Netflix, Youtube, Instagram, Dropbox, Uber, Spotify, Reddit Pinintercest etc...

Django Frame-Work

- Web application framework is a toolkit of all components needed for application development.
- Django is a high-level open source python based frame work which is responsible for rapid development of web application with clean programmatic design with more security.
- Django follows MVC & MVT Design pattern.

Design Pattern

- Design pattern are use described how the data and code segregation should be done while developing application.

There are two type's of design pattern.

MVC - Model View Control

MVT - Model View Templete

MVC

It is a design pattern which responsible for designing an application with separation of data happens in the following format.

MODEL

Model is responsible for writing all the operation's related to the data base.

VIEW

This is a file when we wright the logic by using python programs.
This is used for displaying the content of data base end to the user (FE)

CONTROL

Django framework itself take care of this controlling part.

MVT

MVC + Templates

TEMPLATES

It is used for dealing with HTML files.

MVT is similar to MVC design pattern but in case of MVT design pattern we will be responding an html files as a response to the request sent by the client (user)

Installation of external python packages

Go to command prompt and do the below operation.

pip install packagename # it will install latest version of package.

pip install packagename == Version number # specific version of package.

Make sure that you system is having internet connection while installing python packages.

Note- Before installing any packages first install virtual environments.

pip freeze

It display all packages which you have installed in your system.

Virtual Environment

- Virtual environment is used for deviding a system into number of isolated brand new system.

- Purpose of creating isolated parts is to avoid affecting of download python packages version to the entire system.

Processes to create virtual environment

First we have to install virtual package

pip install virtualenv

create a virtual environment by using below syntax.

virtualenv name of the virtualenvironment.

or

python -m virtualenv name of the virtualenvironment.

- Immediately after creating virtualenv one brand new system will be created
- Brand new system which is created will be having the following files

lib Scripts Environment Configuration.

- We have to enter into that created virtual environment

cd virtualenv name.

- We have to active virtual environment

cd scripts

activate.

- Installing django inside virtual (because django software version should not be affecting entire system)

pip install django

- To turn off our activated virtual environment after use have to run the following command

deactivate.

CD - Current Directory.

Creation of Django Project.

Project is a container which is consisting of many number of features (Application)

Procedure to create a project

- Syntax to create a project in django is

  * django-admin startproject ProjectName

- Immediately after executing above command, one container will be created under the name of projectname.
- Inside the container, we will have 2 files

  Project_Name folder - It is a python package created by-default by django and it is responsible for intergating the application.

  Manage.py file - It is a managerial file responsible for controlling & triggering of the server (starting & stopping of server)

- Information of file present inside the project folder.

__init__.py -

it is a blank python file which is used for making or converting folder with collection of modules into a package.

asgi.py (Asynchronous server Gateway Interface)

asgi is used for creating standard interface between framework, project and web servers

interface

Place or platform by using which people or servers interact with each other

settings.py

It is a file where we will be making all the changes with respect to your project, application, database, static files and templates etc..

urls.py

urls.py file is responsible for performing the process of url mapping

url mapping

It is a phenomenon of connecting or mapping the url with their respective function or class

wsgi.py (Web server Gateway Interface)

wsgi is used for hosting your application or software into the web.

Application

- Application is nothing but the features of project.
- In a single project we can have multiple application.

Creation of application

Syntax for creating an application

* python manage.py startapp app-name

Information of files present inside application

__init__

It is a blank file which is responsible for representing a folder of python modules into package

apps.py

It is a file where we will be doing all the application related configuration.

models.py

It is a file where we create our models (tables)

* models.py creating tables.

admin.py

It is a file which is responsible for registering of created models.

* admin.py registering of created tables.

migrations

It is used for storing the information related to models (tables) of the data base.

views.py

It is a file where we write views (functions or classes) which are responsible for responding for received request.

tests.py

It is a file where we write our test cases which is responsible for testing whether created functions or classes are working fine or not.

Syntax for running the server

```
python manage.py runserver
```

- Immediatly after running the server django creates an default url if there are no errors.
- Now copy the url and search the same in the browser.
- After execution we can stop the server by using control c

Default database of django-

- Django will have sqlite3 database as default database.
- sqlite3 is a light weight data base.
- Immediatly after running the server db.sqlite3 file will be activated inside the project folder.

url (Uniform Resource Locator) -

information about the parts of url

http://127.0.0.1:8000/suffix

url is a combination 4 important things.

protocol:

- It is used specifying the standard rules that we have to follow for data processing and communication.

http:

- It is unsecured and opensource protocol
- We can't process large amount of data.
- used only while developing.

https:

- It is secured and ~~payed~~ paid protocol
- We can process large amounts of data
- Used while developing and after deploying

Insecure

- It is logical address of the server
- As you are running the server in your system django itself creates one virtual address that is 127.0.0.1

Portnumber

- It is address of the channel through which the controller enter into the server
- Default django port number (channel address) is 8000

suffix

suffix is used for navigating to a particular page from a multiple pages

There are 2 types of suffixes-

primary suffix secondary suffix

Extensions to be installed in vs code

Python jinja

code .

It is a command that is used for carrying the current paths content to the visual studio code editor.

Views

Views are callable which take's request and return a response in the form of http

We can return either string or HTML file as an response.

Procedure to connect app to project and create urls. views

- First create project
- now create application

syntax for creating an app

python manage.py startapp app_name

- procedure to register app with project

go to settings.py of project

scroll down until you find INSTALLED_APPS VARIABLE

now register your app by specifying the name as a element of the list.

- Procedure to create views

there are 2 types of views

function based view

class based view

Go to views.py file of application

create a function which is accepting request as first argument.

Inorder to response a string we need HttpResponse function.

so import HttpResponse from django.http

sample example view

```
def function_name(request):
    return HttpResponse('data that we have to respond')
```

- procedural to create urls

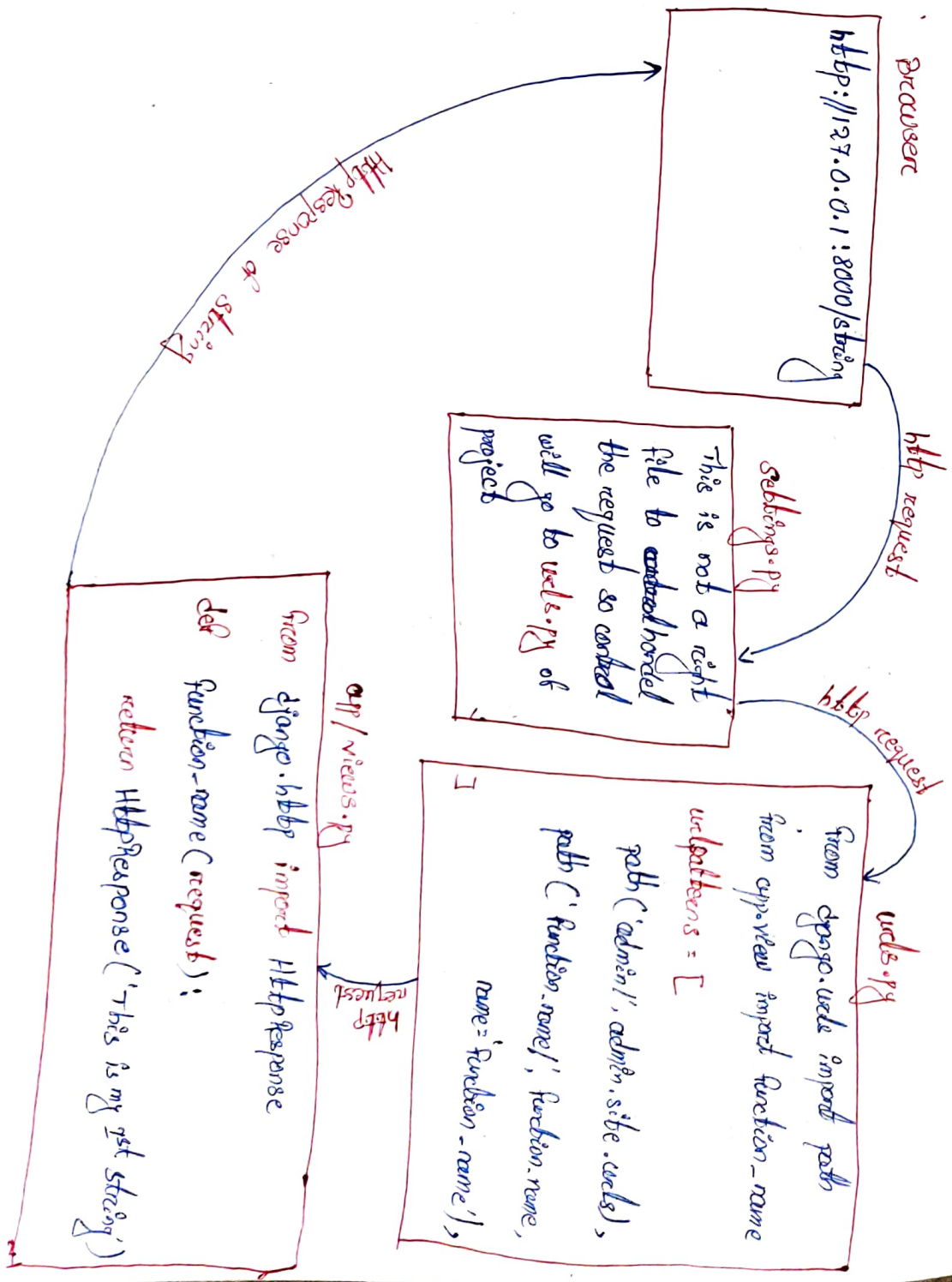
Go to urls.py file of project

we can create urls by using path function

syntax for path function

path('suffix/', address of the function_name, name = 'name of the url')

inside the url patterns list only we have to create urls.



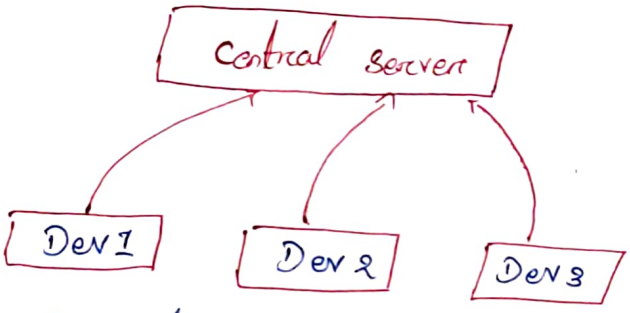
Version Control System

These are used for dealing with the different versions of the software development process.

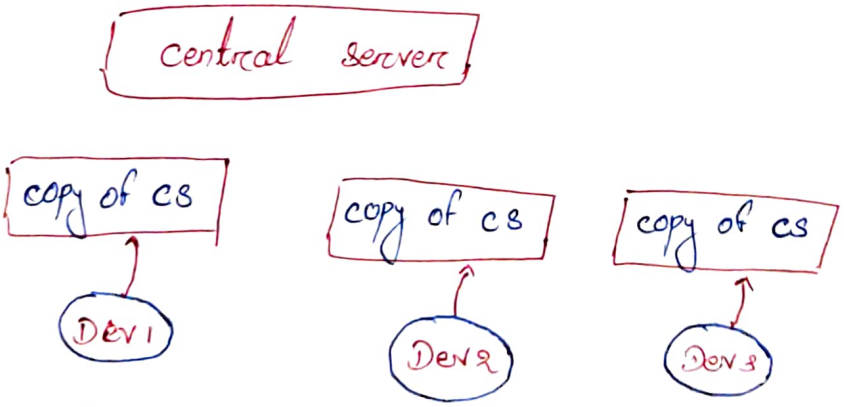
Classification of Version Control System

Generally version control system are classified into two types.

- Centralized Version Control System
- Distributed Version Control System



(Example for Centralized Version Control System)



(Example for Distribut Version Control System)

GIT

It is a distributed version control system which responsible for storing and managing code we are writting.

GITHUB

- Git hub is a cloud based open-source software which used for hosting and managing the git repositories
- Developers use this git-hub for code transmission purpose.

GIT - Repository

It is a folder that we are creating in memory which's is provided by github.com

Procedure to upload project in github

- Go to the browser and search for below url

  http://git-scm.com/downloads
- click on download 2.29.2 version
- search for the url below

  http://github.com
- Signup into github by providing urls details
- After signing up in into the github through browser.
- click on (+) symbol and click on new repository.
- Give a name for the repository and click on create repository
- Go to the git command prompt by carrying path of the file which need to be pushed on git.

Run the following commands

* GIT init

Initalizes a local git-memory inside our local system.

* GIT Add .

It is used to connect the foldees and files which are present in the current cmd path into git memory.

* GIT commit -m 'first commit'

By using above command we can save the connected files in our local git memory.

* GIT branch -m main

It is used for creating one branch with Name main

If you want to change the name we below syntax

git branch -m branchname

* Git remote add origin https://github.com/Sathyayarlanda/first_project.git

this above command is used for creating a repository (folder) inside github.com

* Git push -u origin master

This command is used for pushing the saved connected files of git memory of local system into the github.com

Drawbacks of Generic urls.py in case of multiple Application

- Code complexity and time complexity will be more
- We can't define function with same name in multiple application

Specific urls.py file

- Specific urls is nothing but creating a individual urls.py file inside every application
- To create urls.py file right click on app and chose new file and save as urls.py

Purpose of creating specific urls

- To avoid difficulty in modifying and deleting the urls of particular applications when u have multiple application.

content we need to write in specific urls file

- import path
- import views
- urlpatterns list to be created
- specify value for app_name

url representation of specific urls

Syntax-

http://127.0.0.1:8000/primary suffix/secondary suffix

primary-suffix is used to navigate the control from project urls to app urls

secondary-suffix is used to navigate from app urls to views of application

We use include function to connect specific urls.py with generic urls.py

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

Syntax of path with include

```
path("suffix/", include('app_name.urls'))
```

Sample content of urls.py looks below.

urls.py of project

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

  urlpatterns = [
      path('admin/', admin.site.urls), # app == admin no
      path('app1/', include('app.urls')), # app == app yes
  ]
```

urls.py of app

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

app_name = 'app'

urlpatterns = [
    path('function1/', views.function1, name='function 1'),
]
```

Templates Directory

Templates directory is specifically used to store only html files

There are 2 types of templates directory -

Generic template directory Specific template directory

Generic template directory

If we create template directory inside main project that type of template is known as Generic template directory.

Procedure to create and register templates directory.

Create a project

Create an application and register in settings.py

Create a templates directory by command mkdir templates

Procedure to register templates

Go to settings.py

Create TEMPLATE_DIR variable and store the path of templates as value.

There are three methods how we can add path of a template folder.

Method 1

Create TEMPLATE_DIR variable

copy the path of templates folder

past it as a string value for TEMPLATE_DIR

Change all the single slashes to double slashes

register TEMPLATE_DIR in DIRS key of TEMPLATES

past it as a string value for TEMPLATE_DIR

Change all the single slashes to double slashes

register TEMPLATE_DIR in DIRS key of TEMPLATES

Drawbacks of this method

This method will not work if we are storing the templates directory in different path other than the specified.

Method-2

Concatination of path of BASE_DIR with templates

- As BASE_DIR give the path till the main outer project independent of path where we save it.
- So we can represent path as shown below

```
TEMPLATE_DIR = BASE_DIR / 'templates'
```

Register TEMPLATE_DIR in DIRS key of TEMPLATES

Drawbacks of this method

- This representation of path will not work in all type of operating system because, we use forward slashes for representing path in windows and backword slash in linux.
- So, we go for representation of path by using os module.

Method 3

using os module.

- This is the universal way of representing path
- In real time we use this method only.

Import os

TEMPLATE_DIR = os.path.join(BASE_DIR, 'templates')

Register TEMPLATE_DIR in DIRS key of TEMPLATES

Creating HTML files

- Select and right click on the templates folder
- click on new file
- Give the name of the html file with extension (ex: hi.html)
- Write the content by using html tags.

This is how views.py looks like while we render files

```
from django.shortcuts import render

    def function_name(request):
        return render(request, 'htmlfilename.html')
```

Jinja tagging

Jinja tagging is a modern day tagging (template) language for python developer

OR

- Jinja tagging is a advance level writing tags in html files
- Jinja tags are also known as expression tags or template tags.

Classification of jinja tags

- Based on the operation we are doing with jinja tags they are classified into 2 types
- tags used for printing the data which is just from BackEnd to FrontEnd.
- Tags that are used for performing some operation.

Tag's used for printing

Procedure

- Go to html file represent jinja tag as shown below

  << variable_name >> or << key_name >>

  Example - << mobile >>

- Go to views.py create a dictionary inside the function and pass it as value for the context attribute.

```
def fun_name(request):
    dict = {'key_name': 'value of key'}
    return render(request, 'htmlfilename', context = dict)
```

- Usage of content attribute name is optional

```
def jinja-print(request):
    d = {'name': 'ASHU', 'age': 3}
    return render(request, 'jinja-print.html', content = d)
```

jinja tags that are used for performing some operation:

Wherever we want perform some operation's like ~~inserting~~ inserting images creating hyperlinks, we go for this type of tags.

Representation jinja tags for operation

```
{% expression %}
```

Operations that can done by jinja tag are as follows

- url navigation
- if conditional statements
- looping statements
- insertion of static file like images, css, js
- template inheritance.

conditional statement by jinja tag's

```
{% if condition %}
    statement
{% endif %}
```

If condition syntax

Syntax for if-else condition

<% if condition %>

statement of if block

<% else %>

statement of else block

<% endif %>

Syntax for if-elif-else condition

<% if condition %>

statement of if block

<% elif %>

statement of elif block

<% else %>

statement of else block

<% endif %>

Syntax for nested if-else condition

```
<% if condition %>
    <% if condition %>
        statement of if block
    <% else %>
        statement of else block
    <% endif %>
<% else %>
    <% if condition %>
        statement of if block
    <% else %>
        statement of else block
    <% endif %>
<% endif %>
```

Looping statements

- We can't perform while looping with jinja tags
- We can't perform for loop with range.

Syntax for for loop with collection data type

```
{% for var in CDT %}
    statements of for loop
{% endif %}
```

Url Navigation

- It is a process of navigating from one HTML page to another HTML page by using url mapping names.
- We can achive url navigation by using Anchor tag

Representation of href value for generic url file

```
<a href = "{% url 'name_of_url' %}"> content </a>
```

Representation of href value for specific url file

```
<a href = "{% url 'app_name_value : name_of_url' %}"> content </a>
```

Creation of static folder

- In order to store images, JS properties we use static folder.
- As content inside this will not change so we call this folder as static

We can create static folder in 2 way's

Generic static folder

Specific static folder.

Generic static folder

In this static folder all the images, JS, CSS properties of all applications will be stored in one static folder.

steps to create and register static folder

Create a project

cd project

mkdir static

cd static

mkdir images

mkdir css

mkdir JS

cd ...

Registering generic static folder

Go to settings.py

Create a variable STATIC_DIR and store the path of static as a value

Example - STATIC_DIR = os.path.join (BASE_DIR, 'static')

Create a list-variable with a name STATICFILES_DIRS and do as shown below

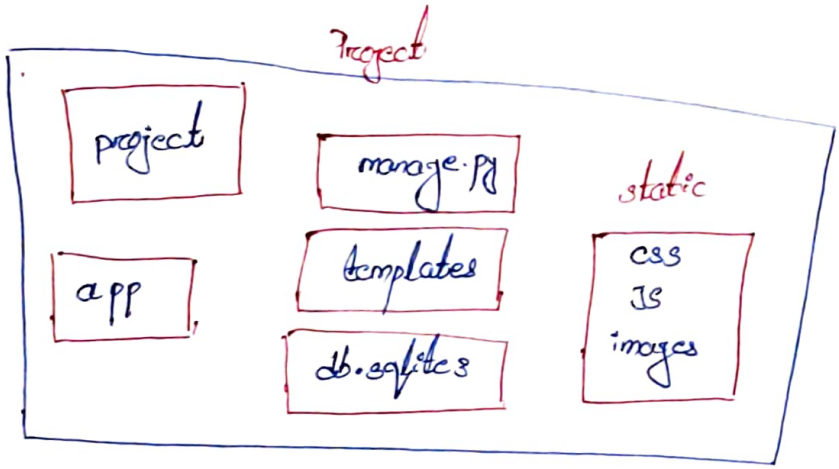
Example- STATICFILES_DIRS = [STATIC_DIR]

steps to check whether static folder connected or not -

python manage.py runserver

http://127.0.0.1:8000/static/images/imagename.extension.

You have to run the above url, if it is displaying the image then url static folder connected successfully.



Insertion of images into webpages

We use img html tag for inserting images

Syntax for inserting image in django is

<img src = "{% static " images/image_name.extension " %}">

- You will get an error because you have not loaded static files into the html file
- So inorder avoid this's load the static file into html by using below command.

{% load static %}

connecting css to html

we use link tag for connecting css

syntax for connecting css file in django is

```
<link href = '{% static "css/cssfilename.extension" %}
```

Specific static folder

In any project css and js properties will be same for all the application

- So we will have css, js properties inside inner projects static folder.
- And we will have images inside the app static folder.

Steps to create and register specific static folders

- Create a project, app
- Create a static folder inside inner project to store css, js
- Again create another static folder inside an app to store images
- Go to settings.py file
  1. Register ur app
  2. Register your templates
  3. Create a variable with a name STATIC_DIR_PROJECTNAME to store the path of projects static folder.

Example -

```
STATIC_DIR_PROJECT = os.path.join(os.path.join(BASE_DIR, 'project'), 'static')
```

- Create a variable with a name STATIC_DIR_APP to store the path of app static folder.

```
STATIC_DIR_APP = os.path.join(os.path.join(BASE_DIR,'app'),'static')
```

- Create a list variable STATICFILES_DIRS to store the above variables

```
STATICFILES_DIRS = [STATIC_DIR_PROJECT, STATIC_DIR_APP]
```

- In order to collect and store all the css, JS and images from specific static folders we need to create one root static directory

~~static~~ - STATIC_ROOT = os.path.join(BASE_DIR, 'static')

- After registering we have to run below command to collect the static files

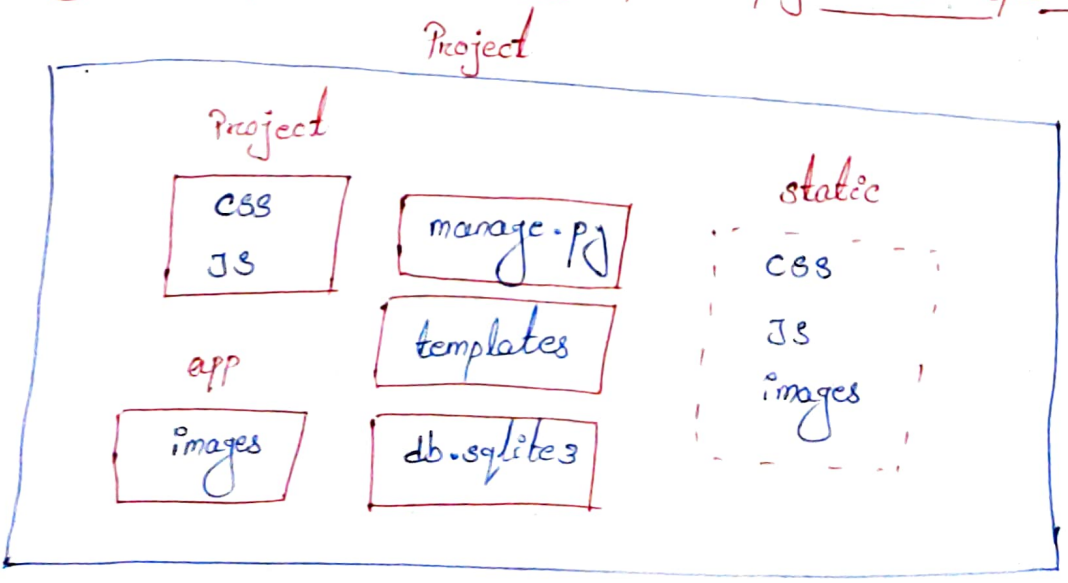
```
python manage.py collectstatic
```

Note

(i) In case of specific static directory we should not add static properties directly into your root static folder.

(ii) If you make any changes in static folders it is mandatory to run collectstatic

Diagramatic representation of files present project incase specific static directory



BOOTSTRAP

- Bootstrap is a css framework
- Bootstrap has some pre-defined classes with which we can add some css features.
- Bootstrap is css framework in which the developers have given default properties for each and every tags.
- We can make a webpage more responsive by using bootstrap.

Procedure to link Bootstrap to HTML files

In two ways we can connect Bootstrap to our html page.

CDN method

Download method

CDN (content delivery network method)

- Go to browser and search getbootstrap.com
- click on get started
- copy the starter template in which they have connected bootstrap css and it's properties past the same in our respective HTML file.

Dis-Advantages of CDN method

- Network connection must be present.
- CDN cache will be cleared if we are not using particular content regularly
- There might be a chance of blocking CDN when you block spam content.

- Go to browser and search getbootstrap.com
- Click on getstarted.
- Copy the starter template in which they have connected css and js properties.

Download Bootstrap to our project

- Go to Bootstrap.com
- click on download. bootstrap download page will be opened
- There will be download button click on it
- Immediately one bootstrap properties ZIP file will be downloaded
- Copy the css properties of zip file and paste it in css folder of your project.
- copy the js properties of zip file and paste it in js folder of your project.

Connecting css & JS properties

We use link tag to connect css properties

We use script tag to connect js properties

Example -

```
<head>
    <link rel = "stylesheet" href = "{% static 'css/bootstrap.css' %}"
</head>
<body>
    <script src = "{% static 'js/bootstrap.js' %}"></script>
</body>
```

MDB

- MDB stands for material design bootstrap.
- MDB has built on bootstrap framework

steps to download MDB4 (Jquery)

- Go to browser and search for below url

  https://mdbootstrap.com/

- click on getstarted.
- scroll down and click on Download bootstrap4 with Jquery zip
- After downloading open the zip file and enter into css and copy

  bootstrap.css mdb.css

  paste into css folder of your project.

- From JS copy below mentioned files bootstrap.js mdb.js jquery.js popper.js

steps to download MDB5 (plain java script)

- Goto browser and search for below url

  https://mdbootstrap.com

- click on getstarted.
- Scroll down and click on download bootstrap5 with plane javascript zip
- After downloading open the zip file and enter into css and copy

  mdb.css mdb.min.css

  paste into css folder of your project.

- From js copy below mentioned files mdb.min.js and paste into js folder of our project.

Template Inheritance

- To inheriting our html page to another html we can do the inheritance.

```
{% extends 'parent.html' %}
```

In parent.html

```
{% block title %}
    <title> parent </title>
{% endblock %}
```

In child.html

```
{% block title %}
    <title> child </title>
{% endblock %}
```

If you want to block body content

Parent.html

```
{% block body-block %}
    content of body
{% endblock %}
```

child

```
<%. block body-block .%>

    content

<%. endblock .%>
```

- We can't have multiple body block in one html page.
- If we want add extra feature in child html page without blck anything in parent's html so go for dummy blck body.

In parents html

```
<%. block body-block .%>

<%. endblock .%>
```

Note

- It is difficult to connect css and Js properties to each and every html files so inorder to avoid that difficulty we have a concept called template inheritance.

Template Inheritance

- It is a process of inheriting the properties of one html file into another HTML file.

Advantage

- It will reduced the code redudancy and increase code reusability.
- With in less time we can build our html content.

- We can use extends jinja tag to inherite the properties.

Syntax

```
<%. extends 'parent_htmlfilename.extension' .%>
```

Example

```
<%. extends 'parent.html' .%>
```

Problems

- If extend then all the parent html content will be copied to child html along with the title.
- So in order to avoid we use jinja block tag's to stop inheriting all the properties.

Jinja tag syntax for blocking title

```
<head>
    <%. block title .%>
        <title> parents.html </title>
    <%. endblock .%>
</head>
```

Jinja tag for blocking body content

```
<body>
    content we need to inheritance
    <%. block body_block .%>
        content which not to be inherited
    <%. endblock .%>
</body>
```

Note

- We can't have more than one body-block in a single html file
- It is mandatory to have empty body block in parent html, in case if you want inherit all the properties of parent into child and add a new properties to child

N-TIRE Architecture

- In this type of architecture we will be having more than 2 layers that means our application will be having the following ends or phases

FRONT END

BACK END

DATABASE END

- In development language we call the database end as model layer

Models

- Popularly we can store the data in 2 formats

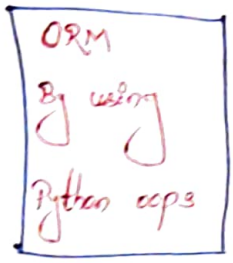
TABLES

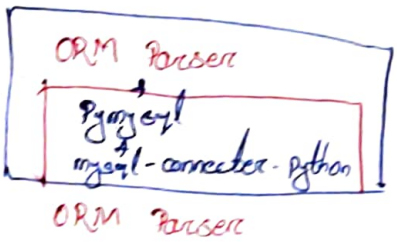
DOCUMENT FORMATS

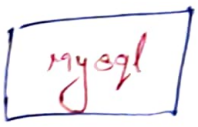
- In our django framework we will store the data in the form of tables
- As it is relational DBMS we have to use language like mysql, postgresql, sql etc to manipulate the data.
- As it is difficult to learn sql or Mysql or PostGresql, django has come up with the concept of ORM to manipulate the data

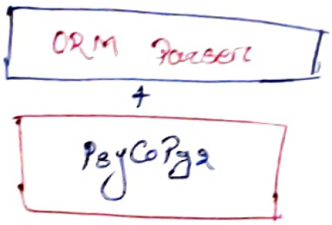
ORM (Object Relational Modelling)

- It is a concept which responsible for writing database queries in the form of python object Oriented Programming
- We can use any type of database.
- By default django is provided with sqlite3 database
- We can connect other databases instead of sqlite3 just by pip installing it and making some changes in the database section of our settings.py.









Creation of database table

- In order to create a table by using ORM we have to follow below consideration

class in ORM ⟶ Table in data base

class variable in ORM ⟶ Column in data base

Object in ORM ⟶ Row of data in data base

steps to create models

- Go to models.py of app

- Each and every model class must be inherited from model class as shown below
- Create a class which represent as table_name with a syntax given below.

```
class tablee/model_name (models.model):

    class_member = models.fieldname (argument)
```

Above created

- class memory is used to represent column_name
- File name is used to represent data type.
- Arguments will be acting as constraints
- We have to create a method with a name __str__ which is used for returning a string.

Creating following Table

Table_Name	Topic	WebPage	AccessRecord
column_name	Topic_Name	Topic_Name Name URL	Name Data author

content of models.py file

```
from django.db import models

# Create your models here

class Topic(models.Model):
    topic_name = models.CharField(max_length=30, primary_key=True)
    def __str__(self):
        return self.topic_name

class WebPage(models.Model):
    topic_name = models.ForeignKey(Topic, on_delete=models.CASCADE)
    name = models.CharField(max_length=50)
    url = models.URLField()

class AccessRecord(models.Model):
    name = models.ForeignKey(WebPage, on_delete=models.CASCADE)
    date = models.DateField(auto_now=True)
```

Giving access of model to admin

- Go to admin.py file of app
- import all the models into admin.py
- Register the models with below syntax

admin.site.register(model_name)

Example

admin.site.register(Topic)

storing the models into our database

- Go to command prompt
- python manage.py migrate

  (This command is used to check all the imported models are satisfying all the rule and regulation or not.)
- python manage.py makemigrations

  (This command is used to converting all the object oriented class that we have created into database language)
- python manage.py migrate

  (After running (python manage.py makemigration) again we have to run (python manage.py migrate) This time it will create actual into the database after conversion by ORM parser.)

Creating Super User

- In order to create superuser we have to run the following command in cmd.

python manage.py createsuperuser

- After running the command in cmd it will ask for

  Username :

  email :

  password :

  Re-enter password :

- We have to conform the details so press Y with this superuser will be created.

- Runserver and go to browser and search for below url

  http://127.0.0.1:8000/admin/

- Admin interface will be opened give username and password.

Operation we can do with the tables

Insertation of data Retrieving of the data

Deleting the data Updating the data.

Insertation of Data

We can insert the data into our models in 3 way's

- By creating an object and passing the values as key-values
- By create method
- get_or_create method.

creating an object to insert data.

syntax

- Objectname = classname(columnname = value, columnname2 = value...)

- Inorder to save the inserted the data we have to make utilize of save() method of Model class

```
objectname.save()
```

changing cmd into interactive shell

- Run the below command to enter into interactive console. ~~python~~

```
python manage.py shell
```

- Run the below command to come out of interactive console

```
exit()
```

Inserting data by using create method

syntax

```
objectName = classname.objects.create(columnname = value, col2 = value)
```

- Inorder to save permanebtly use save method

```
objectname.save()
```

- When you use create method to insert the data it will return a instance directly.

Inserting data by using get_or_create method

Syntax

```
Objectname = classname.objects.get_or_create(colname = value)[0]
```

- In order to save use save method

```
Objectname.save()
```

- When you use get_or_create method to insert the data it will return a tuple

```
(instance, booleanflag)
```

Note

get_or_create method is a universal method which we will be using to insert data in real time.

Retrieving of the data

In many ways we can retrieve the data from tables by using orm

Syntax for retrieving all the data

```
modelname.objects.all()
```

Process to display the retrieved content

- create one html file
- Go to views import all the models
- create a variable and store the values of queryset

Example

variablename = modelname.objects.all()

- Send the query set to the front end by using context
- Use jinja tags to display the context data.

Syntax for retrieving single data or particular data

There are 2 methods by using which we can retrieve particular data

get filter

get method

- Based on condition we can retrieve the data
- It will return directly the object.

Syntax

modelname.objects.get(columnname = value)

Drawbacks

- value must be a primary key

Get through an error

- If none of the rows are satisfying the given condition
- If more than one row is satisfying the condition

filter method

It will return queryset in the form of List.

syntax

```
modelname.objects.filter(columnname = value)
```

Note

It is ideal method by using which we will retrieve the data from the models.

For all data

```
modelname.objects.all()
```

For specific data

```
modelname.objects.get(cond) → Only for Primary key.
modelname.objects.filter(cond) → Only for any column.
```

Example

Scenario	get	filter
One row	It returns object directly	Queryset with list of objects which satisfy condition.
more than one	Error	List of object queryset.
No rows	Error	Empty Queryset.

Ordering of retrieved data-

Ascending order

```
modelname.objects.order_by('columnname')
```

Descending order

```
modelname.objects.order_by('-columnname')
```

Arranging an order based on length

First import length function from django.db.models.functions

```
From django.db.models.function import length
```

For Ascending order

```
modelname.objects.all().order_by(length('columnname'))
```

For Descending order

```
modelname.objects.all().order_by(length('columnname'))
```

Exclude method

By using exclude method we can retrieve the rows which are not satisfying the condition.

```
modelname.objects.exclude(columnname = value)
```

Field lookups

- These are used for achieving where conditions more precisely.
- Field lookups start with double underscore.

```
column name __ lookupname = value.
```

- This lookups must written inside the filter method

```
modelname.objects.filter(column name __ lookupname = value)
```

Different types of field lookup

- `__gt` → Greater than (>)
- `__gte` → Greater than equal to (>=)
- `__lt` → less than (<)
- `__lte` → less than equal to (<=)

Date format

```
'YYYY-MM-DD'
```

Month lookup

It is used for comparing only the months of a data

```
columnname__month = 'value in number'
```

Year lookup

It is used for comparing only year of a date

columnname__year = 'value in number'

Day lookup

It is used for comparing only the day of the date.

columnname__day = 'value in number'

* Like operator functionality can be achieved by using below methods.

startswith

It is used for checking whether specified string is starting with specified value or not.

modelname.objects.filter(columnname__startswith = value)

endswith

It is used for checking whether specified string is ending with specified value or not.

modelname.objects.filter(columnname__endswith = value)

In

It is used for checking for multiple value of same column.

```
modelname.objects.filter('columnname__in=(value1, value2, value3))
```

Contains

It is used for searching specified character are present inside the string or not.

```
modelname.objects.filter(columnname__contains=value)
```

Regex

It is used for comparing by using the patterns

```
modelname.objects.filter(columnname__regex=r'pattern')
```

Note- r represent .Raw. Pattern.

Q objects

1- Q object is mainly used for encapsulating key word arguments
- We use Q objects whenever we are dealing with complex conditions or multiple conditions.
- In order to utilise Q objects we have to import it first from

```
from django.db.models import Q
```

- We can combine more than one query set by using &/| operators.

  & → stands for AND operator

  | → stands for OR operator.

- Then new query will be emerged by combining the represented queries

Syntax

```
modelname.objects.filter(Q(condition1) & Q(condition))
```

Updating the records

In 2 way's we can update the records.

Update_or_create method Update method.

Update method

Syntax for updating a particular row

```
modelname.objects.filter(condition).update(columnname = value)
```

syntax for updating entire table info

```
modelname.objects.all().update(columnname = value)
```

Note-

- If one row satisfies it will update that particular row
- If multiple rows satisfies it will update multiple row
- If no row satisfies the condition ; It will not perform any operation.

- While modifying foreign key column we need to provide value present in the parent table.

Update_or_create

Syntax

```
modelname.objects.update_or_create(columnname = value, ..., defaults={'column' : 'value'})
```

Note

- If one row satisfies it will update that specific row
- If multiple row satisfies it will throw Error. (We can't modify multiple row by using update_or_create method.)
- If no row's satisfies it will create a new row.
- While modifying foreign key column we need to provide instance of parent table.

Deleting record's from the models

- We use delete method in order to delete data from the tables.

Syntax for deleting a particular record

SQL query

```
delete from table_name where condition;
```

ORM code

```
modelname.objects.filter(condition).delete()
```

← No argument.

Syntax for deleting an entire data from a model.

SQL query

```
truncate table table_name;
```

ORM code

```
modelname.objects.all().delete()
```

FORMS

In order to take input from the user we will use forms

In django this forms are classified into 3 types

Normal HTML forms Built-in django forms Model forms

Normal HTML forms

By using form input textarea tags of html we will create forms

Sample example

```
<form action = 'urlname/filename' method = 'GET/POST'>
    <input type = 'text'> name </input>
</form>
```

Action attribute

By using this attribute we can carry the submitted data to another url or html file.

syntax

```
action = '{% url "urlname" %}'
```

Method

It is used to specify how we have to carry the submitted form data there are 2 way's how we can carry the data.

GET method

POST method.

GET method

- It is used when we are using input element inorder to search some data
- Large amount of data can't be transferred.
- It is unsecure method.

POST method

- By using this method we can carry the data inorder to store into the data base. large amount data can be transferred.
- It is secure method.

csrf_token

It is used for sending the form data more securely

syntax

```
{% csrf_token %}
```

Process Happens after submitting the data

- By default same URL function will be called.
- POST method will be activated once you submit the data.
- submitted data will be collected under the dictionary
- The name of the dictionary is request POST

sample of submitted data

```
request.POST = {'Un': ashu, 'PW': 'nikky'}
```

Procedure to access the form input elements data

sample html file

```
<form method = 'POST'>

    <label for = 'username'> Username </label>
    <input name = 'User' id = 'username'>

    <label for = 'password'> Password </label>
    <input name = 'pass' id = 'password'>

</form>
```

In 8 way's we can access the data from submitted form

- By using the key name.

```
request.POST ['keyname']
```

- GET method

We use get method whenever we accessing single data.

Syntax

```
request.POST.get('keyname')
```

- getlist method

We use this method inorder access multiple input output format is a list.

```
request.POST.getlist('keyname')
```

Inserting the values into the models by taking values through front-end

views.py

```
def insert_web(request):
    topics = Topic.objects.all()
    if request.method == 'POST':
        topic_name = request.POST.get(TOPIC)
        name= request.POST.get('name')
        url = request.POST.get('url')
        to = Topic.objects.get_or_create(topic_name = topic_name)
        to.save()
        wo = Webpage.objects.get_or_create(name = name, topic_name =
                                           to, url = url)
        wo.save()
    # return HttpResponse('one record has been added')
    return render(request, 'display_web.html')
```

display_web.html

```
<form method='POST'>
    {% csrf_token %}
    {% if topic %}
        <select name="topic" id="topic">
            {% for to in topics %}
                <option value="{{ to.topic }}">{{ to.topic }}</option>
            {% endfor %}
        </select>
    {% endif %}
    <label for="name">Name</label>
    <input type="text" name='name' id='name'><br>
    <label for='url'>Url</label>
    <input type="text" name='url' id='url'><br>
    <input type="submit" value='submit'>
</form>
```