# Environment and setup

* 1. Webframework.
  2. Virtual environment
     1. To create a venv <conda create --name venv python=3.5>
     2. OR <conda create --name venv django=XX>
     3. To activate <conda activate nameOfVenv >
     4. To deactivate <conda deactivate nameOfVenv >
     5. All envs on machine <conda info --envs>
     6. Install django in venv as <conda install django>. This will also install a command line tool called django-admin

1. Creating first project
   1. Ensure your venv is activated
   2. Cd to the folder where you want the project to be created
   3. <django-admin startproject venvproj>. This will create folder [venvproj] and under it another folder [venvproj] and a manage.py file
      1. Inside folder venvproj there are below files
         1. \_\_init\_\_ <because of its special name, this directory is treated as a package>
         2. File asgi
         3. File settings <project settings are stored>
         4. File urls <urls>
         5. File wsgi <web server gateway. Helps to deploy app to production>
2. Creating first app under the first project <venvproj>
   1. Cd at the folder where you have manage.py file in project folder [venvproj ]
   2. STEP1. Run command <python manage.py startapp venvapp> . This will craete a subfolder [venvapp] that as below files
      1. Folder migrations .. it is \_\_init\_\_
      2. \_\_init\_\_ <because of its special name, this directory is treated as a package>
      3. File admin <to register models with admin and use admin interface>
      4. File apps <application specific configurations>
      5. File models <application’s data models, ER>
      6. File test <series of functions to test code>
      7. File views <accepts request and return responses>

* 1. STEP2 we need to let djnago know venvapp exits, so it needs to be added in project’s setting file

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'venvapp',

]

* 1. STEP3. Next step we want to do is to create a view in views.py file.

views.py

# Create your views here.

"""

Next step that we want to create is actually a view. Our first view sends back simple text.

Each view for this application is going to exist within that views.py file as its own individual function. In this instance we just created one view called index with that index function.

Each view is also going to take at least one argument. The http request object, since it lives inside django HTTP module, by conventionwe call it "request".

"""

def index(request):

HttpResponse("HttpResponse Returns this")

* 1. STEP4. We now need to map this view to urls.py file. To do this open the project’s urls.py file and modify as below

from django.contrib import admin

from django.urls import path

from django.conf.urls import url #added by me

from venvapp import views #added by me

urlpatterns = [

path('admin/', admin.site.urls),

url(r'^$',views.index, name='index'), #added by me

]

Now <python manage.py runserver>. It should show output.

* 1. Now we will see URL mapping
     1. Make modifications in urls.py at project level

from django.contrib import admin

from django.urls import path

from django.conf.urls import url #added by me

from venvapp import views #added by me

from django.conf.urls import include ## added for URL Mapping

urlpatterns = [

path('admin/', admin.site.urls),

url(r'^$',views.index, name='index'), #added by me

url(r'^venvapp/', include('venvapp.urls')), ## added by me for URL Mapping

]

* + 1. Create below new urls.py file in venvapp

from django.conf.urls import url

from venvapp import views

urlpatterns = [

url(r'^$', views.index, name= 'index'), ## added by me for URL Mapping

]

* 1. Now we will Add TEMPLATE
     1. Create a directory at same level as manage.py and call it <templates>
     2. Create a variable <TEMPLATE\_DIR> in setting file and import os also.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

import os

TEMPLATE\_DIR= os.path.join(BASE\_DIR,'template')

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [TEMPLATE\_DIR,],

* + 1. Create a new index.html file inside the template folder as below

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title> Tilte </title>

</head>

<body>

<h3>Index.html</h3>

{{ insert\_me }}

</body>

</html>

* + 1. Open views file and change as below

from django.shortcuts import render

from django.http import HttpResponse

def index(request):

my\_dict = {'insert\_me':'The content from key insert\_me'}

return render(request, 'index.html', context= my\_dict)

* + 1. Now if you run <python manage.py runserver> , it will show index.html content
  1. STATIC
     1. In the level same as manage.py create two new folders called <static/images>
     2. Edit settings file and add a new variable called

TEMPLATE\_DIR= os.path.join(BASE\_DIR,'templates')

STATIC\_DIR = os.path.join(BASE\_DIR,'static') #this is added in this step

* + 1. Now go down the setting file and you will see a “predefined variable” for static called <STATIC\_URL>. We want to add another variable that takes a list of static directory paths.

STATIC\_URL = '/static/' # this already exists… below is added

STATICFILES\_DIRS =[

STATIC\_DIR,

]

* + 1. Place a jpeg file in this folder. Now <runserver> and you can see this image via <<http://127.0.0.1:8000/static/images/image1.jpg>>
    2. Now we want to inject this image via template tagging into html file. Go to <E:\Anupam\GITProjects\Django\venvproj\templates\venvapp> where index.html file is located and edit as below. Second line should be {% load static %}.

Beow is the index.html file

<!DOCTYPE html>

{% load static %}

<html>

<head>

<meta charset="utf-8">

<title> Tilte </title>

</head>

<body>

<h3>Index.html</h3>

<img src="{% static 'images/image1.jpg' %}" alt="oh ohhh">

</body>

</html>

* 1. CSS as STATIC file
     1. Under static folder create another folder called <css> and place a css file called <mystyle.css>

h3{

color: red;

}

* + 1. Now change the index.html file to include this css file as below

<!DOCTYPE html>

{% load static %}

<html>

<head>

<meta charset="utf-8">

<title> Tilte </title>

<link rel="stylesheet" type="text/css" href="{% static 'css/mystyle.css' %}"/>

</head>

<body>

<h3>Index.html</h3>

<img src="{% static 'images/image1.jpg' %}" alt="oh ohhh">

</body>

</html>

* + 1. Now rerun the server

1. Level2
   1. MODELS
      1. Open models.py file of app venvapp. Models are classes and each class is like TableName of database.

from django.db import models

# Create your models here.

class Topic(models.Model):

topic\_name = models.CharField(max\_length = 264, unique= True)

def \_\_str\_\_(self):

return self.topic\_name

class WebPage(models.Model):

topic = models.ForeignKey(Topic,on\_delete=models.CASCADE,)

name = models.CharField(max\_length = 264, unique= True)

url = models.URLField(unique=True)

def \_\_str\_\_(self):

return self.name

class AccessRecords(models.Model):

name = models.ForeignKey(WebPage,on\_delete=models.CASCADE,)

date = models.DateField()

def \_\_str\_\_(self):

return self.date

* + 1. Now register the changes to your application. <python manage.py makemigrations venvapp>.
    2. Run migrate one more time <python manage.py migrate>
    3. Now our models are ready and connected to sql database that django created.
    4. SHELL COMMANDS. Now to interact with the models one basic way is to use shell commands as in step below where we will create some test data using shell command
       1. Run <python manage.py shell>
       2. To import Topic model and check all records it has run <from venvapp.models import Topic> .. Then <print(Topic.objects.all())> . This gives empty queryset at this stage
       3. Now to create one record and save it
       4. Run <t= Topic(topic\_name= “Socal Network”)>
       5. Run <t.save()>
       6. Run <quit()>
    5. ADMIN INTERFACE. We can use Admin Interface to add records
       1. Use use Admin interface, we need to register the models to admin.py
          1. Open admin.py and modify as below

from django.contrib import admin

from venvapp.models import Topic, WebPage, AccessRecords

# Register your models here.

admin.site.register(Topic)

admin.site.register(WebPage)

admin.site.register(AccessRecords)

* + - * 1. Create a superuser

Run <python manage.py createsuperuser>

User anupam pwd anupam

* + - * 1. Now we can login to admin interface and add records
    1. MTV (Models Templates Views)
       1. 5 steps
          1. Step1. Import models in views.py
          2. Step2. In view query the model for data we need
          3. Step3. Pass results from model to template
          4. Step4. Edit template so that it is ready to display data from model
          5. Step5. Map a URL to this view
       2. Step1 Import models in views.py
          1. Open views and add <from venvapp.models import Topic, WebPage, AccessRecords>
       3. Step2 In view query the model for data we need
          1. In the view make changes as below

from django.shortcuts import render

from django.http import HttpResponse

from venvapp.models import Topic, WebPage, AccessRecords

def index(request):

webpage\_list = AccessRecords.objects.order\_by('date')

date\_dict = {'acc\_rec':webpage\_list}

return render(request, 'venvapp/index.html', context= date\_dict)

* + - 1. Step3 Open html
         1. Make modifications as below

<!DOCTYPE html>

{% load static %}

<html>

<head>

<meta charset="utf-8">

<title> MTV </title>

<link rel="stylesheet" type="text/css" href="{% static 'css/mystyle.css' %}"/>

</head>

<body>

<h3> MTV </h3>

<h4> Accee Rec </h4>

<div class="Level2MTV">

{% if acc\_rec %}

<table>

<thead>

<th>Site Name</th>

<th>Date Accessed</th>

</thead>

{% for a in acc\_rec %}

<tr>

<td>{{ a.name }}</td>

<td>{{ a.date }}</td>

</tr>

{% endfor %}

</table>

{% else %}

<p>No Access Records Found</p>

{% endif %}

</div>

</body>

</html>

* + - * 1. Run <python manage.py runserver> and you should see html with records rendered