

Chapter 9 – Creating a Django based Basic Web Application

What is Django?

- It is a free and open source web application framework that can be used along with Python to develop dynamic websites. (*Dynamic websites are those which update content depending upon the users' need or a specific area*).



- Django web framework is managed and maintained by an independent and non-profit organization named **Django Software Foundation(DSF)**.
- Other Web Frameworks :
 - Ruby on Rails – Ruby
 - Express –js – javascript

Why Django?

- 1. Secure** – It offers code and enables protection against most of security vulnerabilities.
- 2. Versatile** – Almost any type of website(news site, e-commerce site, a social networking site etc.) can be built using Django.
- 3. Portable** – Since Django is Python based, the websites developed in it can run on any platform like Linux, Windows or Mac OS.
- 4. Easy to maintain** – Django code is easy to reuse and easily maintainable.

How Web, Websites and Web-applications work?

- Web works in the form of **client-server architecture**. Your **web browser** acts as a client program (**front-end**) and the **web server** with which it interacts is the **server (the backend)**.
- The client(web browsers) work on the web mostly with HTTP (hypertext transfer protocol). The clients make an **HTTP request**, which the (web) server responds to in the form of a response called **HTTP response**. The client make these **two types of HTTP requests**:
 - (i) HTTP GET request** – It refers to a way of retrieving information from a web-server using a given URL over web.
 - (ii) HTTP POST request** – It is a way to send data to the server, (e.g. data filled in an online form such as student information, file upload etc.) using HTML forms.

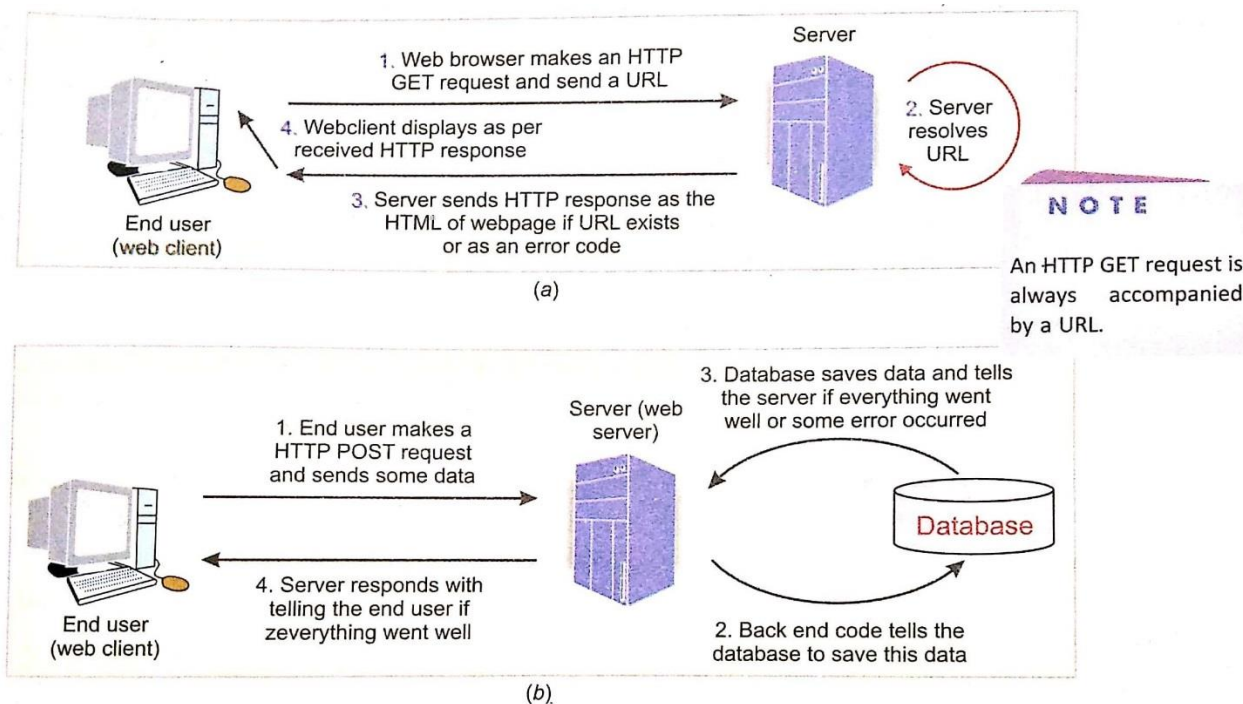


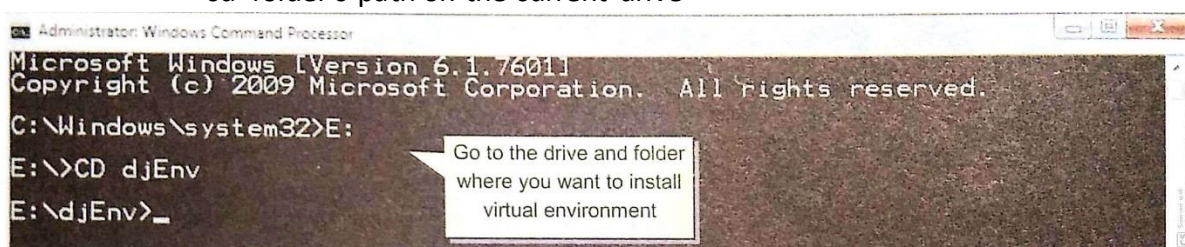
Figure 9.2 (a) GET request (b) POST request.

Installing Django (in Virtual Environment / Virtualenv)

- **Virtualenv** – It is a useful tool which creates **isolated Python environments** that take care of interdependencies and let us develop different applications in isolation. Each isolated environment has different set of libraries unlike a common global set of libraries.

Steps to Install Django

1. Start **cmd**.
2. Go to the drive and folder where you want to install virtual environment for Django. We have created folder namely **djEnv** on **E:** drive for this purpose. Next go to the folder where you want to install your virtual environment. Use command **CD** to change folder, i.e.
`cd<folder's path on the current drive>`



3. Once in the desired folder, type the command:
`pip install virtualenv`
4. Next you need to activate virtual environment by giving **activate** command. Since it lies in folder **venv\Scripts** under current directory, the command we shall type will be :
`venv\Scripts\activate`
5. Once activated, you can install latest version of Django by giving command:
`pip install django`


```

Administrator: Windows Command Processor
E:\djEnv>pip install virtualenv
Requirement already satisfied: virtualenv in C:\Program Files\Python37-32\lib\site-packages (16.3.0)
Requirement already satisfied: setuptools in C:\Program Files\Python37-32\lib\site-packages (from virtualenv) (40.8.0)
Virtual environment installed in current folder.

E:\djEnv>virtualenv venv
Using base prefix 'c:\program files\python37-32'
New python executable in E:\djEnv\venv\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
Virtual environment activated

E:\djEnv>venv\Scripts\activate
(venv) E:\djEnv>pip install django
Collecting django
Using cached https://files.pythonhosted.org/packages/2a/9e/84c2914aace7930ecccbf3b7b98ac9fe584fda72807f5f3fb/Django-2.1.6-py3-none-any.whl
Collecting pytz (from django)
Using cached https://files.pythonhosted.org/packages/d2/43/98a7cd85cc7b9a75a490570d5a90c57622d34/pytz-2018.9-py2.py3-none-any.whl
Installing collected packages: pytz, django
Successfully installed django-2.1.6 pytz-2018.9
Django installed in activated virtual environment.

(venv) E:\djEnv>
(venv) before the prompt means virtual environment is active

```

Test-run Django Development Webserver

- This activity is carried out to test that if we have successfully installed Django and its built-in web server is properly working on our computer.

Steps:

1. In the active virtual environment, type following command in front of the prompt :
`django-admin startproject <projectname>`

e.g. `django-admin startproject testproject`

```

(venv) E:\djEnv>django-admin startproject testproject
(venv) E:\djEnv>

```

2. It will create a directory/folder by the name **testproject** under the current folder. To confirm type command DIR in front of the command prompt and press Enter.

```

Administrator: Windows Command Processor
E:\djEnv>venv\scripts\activate
(venv) E:\djEnv>django-admin startproject testproject
(venv) E:\djEnv>dir
Volume in drive E is Bux
Volume Serial Number is A40C-4E76

Directory of E:\djEnv

02/22/2019  07:36 PM    <DIR>
02/22/2019  07:36 PM    <DIR>
02/16/2019  05:32 PM    <DIR>
02/08/2019  06:33 PM    <DIR>
02/08/2019  06:43 PM    <DIR>
02/16/2019  06:05 PM    <DIR>
02/08/2019  06:45 PM    <DIR>
02/08/2019  06:43 PM    <DIR>
02/22/2019  07:36 PM    <DIR>
02/11/2019  06:25 PM    <DIR>
               0 File(s)              0 bytes
               10 Dir(s)  326,097,596,416 bytes free
(venv) E:\djEnv>

```

You will find a folder having the project name you specified

3. Now change to this folder, your project's folder, using command CD, i.e. give command
cd <projectname>
4. Once you changed the current folder to your project's folder, type following command in front of the prompt to run built-in webserver:
python manage.py.runserver

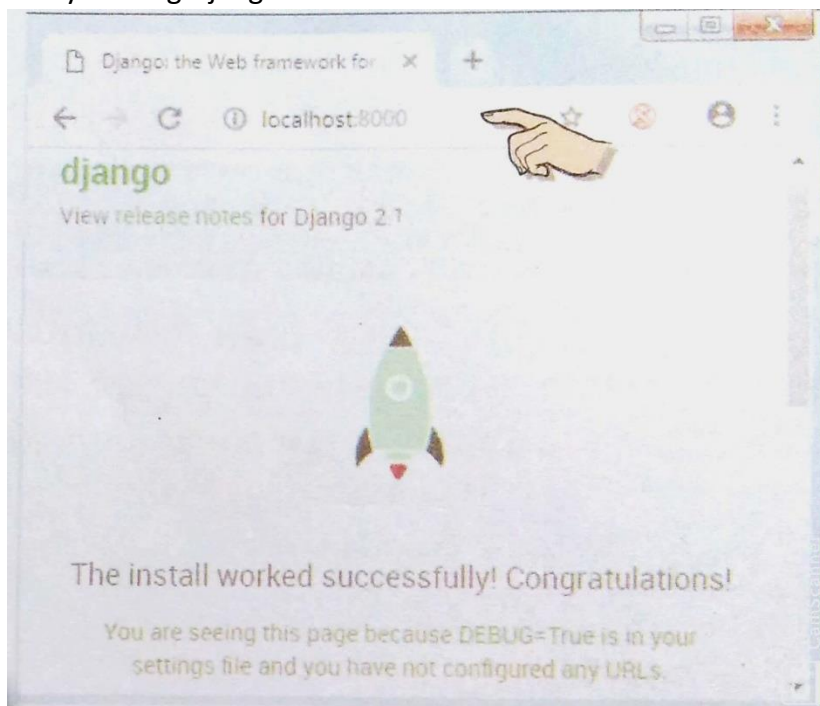
```
(venv) E:\djEnv>cd testproject
(venv) E:\djEnv\testproject>python manage.py runserver
Performing system checks...
System check identified no issues (0 silenced).
You have 17 unapplied migration(s). Your project may not work properly until you
apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
February 22, 2019 - 19:43:53
Django version 2.1.6, using settings 'testproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

5. Finally, open your web browser window and type either of the following commands in the address bar:

localhost : 8000
127.0.0.1 : 8000

127.0.0.1 is the IP address of your local host and 8000 is the default port on which it connects to Django webserver.

If your browser shows following screen – it means that you have successfully installed Django and have successfully running Django webserver.



Django Basics and Project Structure

1. What are Project and App in Django?

- A **Project** refers to an entire application.
- An **app** is a sub-module catering to one part of the project.

e.g.

if you are creating an **online shopping store** (project), then its **apps** can be :

- Customers – It will manage the profiles of customers.
- Products – It will manage the products' details.
- Shopping-cart – It will manage the details of selected products chosen for shopping.

2. Creating a Project in Django

Steps:

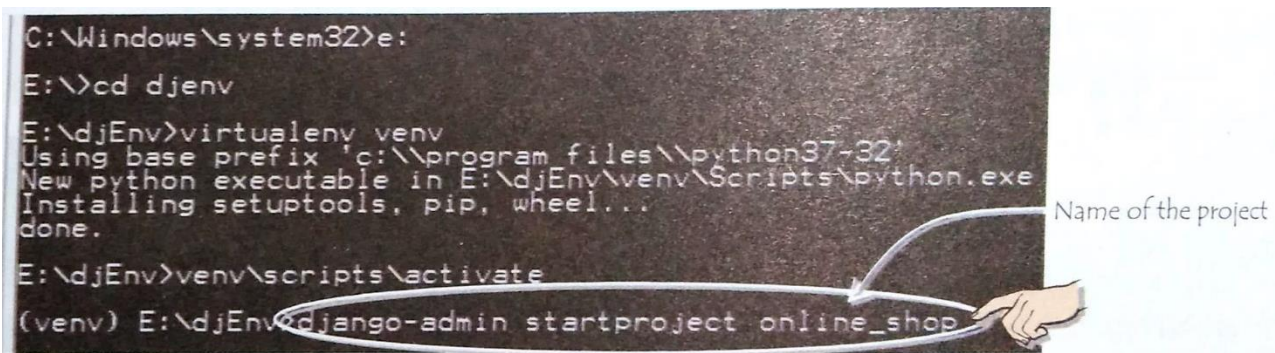
(i) Start Virtual environment as explained earlier.

(ii) On the command prompt type command:

```
django-admin startproject <projectname>
```

e.g.

if your project name is **online_shop**, you will write following command on the command prompt.



```
C:\Windows\system32>e:
E:\>cd djenv
E:\djEnv>virtualenv venv
Using base prefix 'c:\program files\python37-32'
New python executable in E:\djEnv\venv\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
E:\djEnv>venv\scripts\activate
(venv) E:\djEnv>django-admin startproject online_shop
```

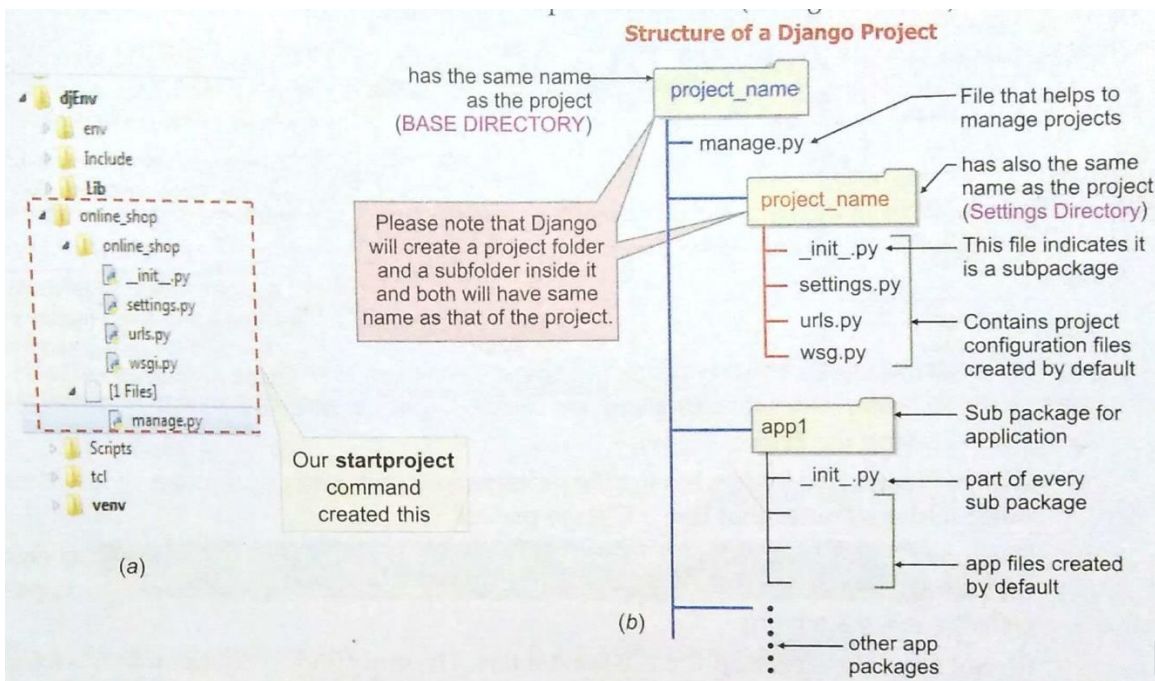
Name of the project

(iii) When you create a project with **django-admin startproject** command, it will create a folder having the same name as that of the project in the current folder. This newly created project folder will have some files and a folder preloaded in it – all because of Django framework.

3. Structure of a Django Project

When you create a Django project with **startproject** command, Django automatically creates a folder bearing the same name as the project. This folder has *some folders and files preloaded* in it. The project folder has another folder in it, also having the same name as that of the project and a file namely **manage.py**.

e.g.



4. Folders Having the Project Name

➤ Files in Outer Project name folder (the base directory)

(i) **manage.py** – used to *manage the project* and perform some *administrative tasks* such as running built-in-server and many others.

➤ Files in Inner Project name folder (the subfolder)

(i) **_init_.py** – This is a file inside every package/sub-package of Python.

(ii) **settings.py** – store configuration settings for your Django project.

(iii) **urls.py** – stores information about the locations where URLs of your Django project have been declared.

(iv) **wsgi.py**

5. Creating Apps of a Django

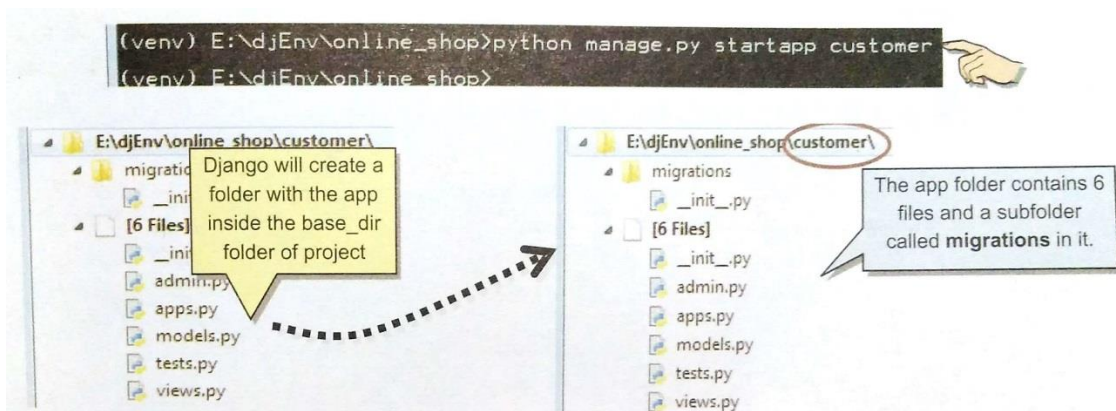
- Individual app can be created by issuing the following command while inside the project folder where **manage.py** resides:

```
python manage.py startapp <appname>
```

e.g.

if we want to create app **customers** inside our project **online_shop**, we shall first go inside the outer project folder **online_shop** (because **manage.py** resides in it) and then issue command:

```
python manage.py startapp customers
```



Understanding Django Project Structure

- Django projects are based on a software architecture that separates:
 1. Data being managed(the *model*)
 2. Presenting/showing to user (the *template*)
 3. Handling Logic (the *view*)

This software architecture was called **MVC (Model View Controller)** architecture and now it is called MVT (Model View Template) architecture.

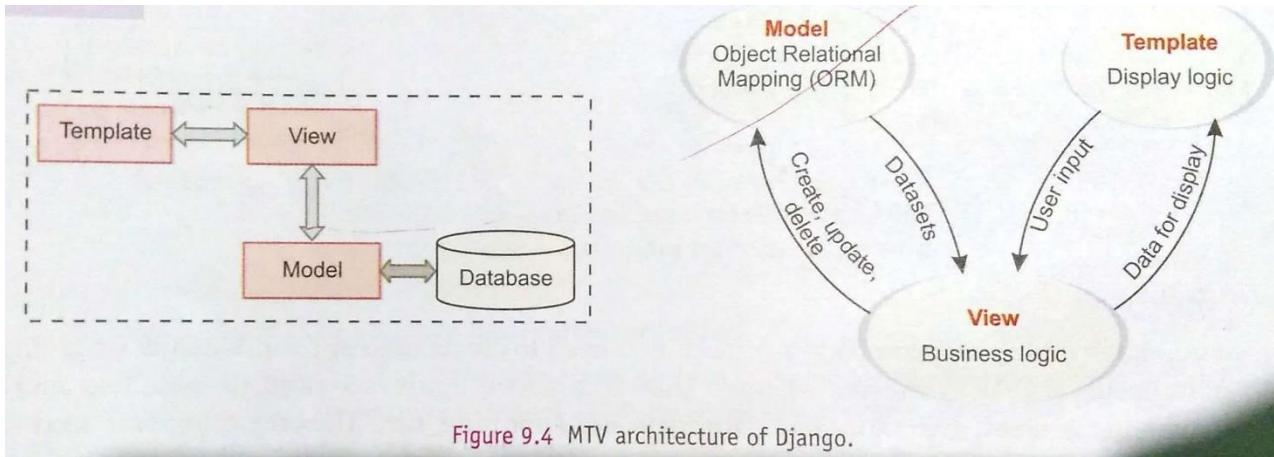


Figure 9.4 MTV architecture of Django.

Steps to Create a Basic Django Web Application

1. Creating Models

- Models in Django refer to the way data is stored and processed.
- Model in Django map to tables in database just like SQL. These models are created through **Object-Relational Mapping (ORM)** which maps to the underlying database.
- You can define data model for your app through following syntax:

```
class <modelname>(models.Model) :  
    <fieldname> = models.<fieldtype>  
    <fieldname> = models.<fieldtype>  
    .  
    .  
    .
```

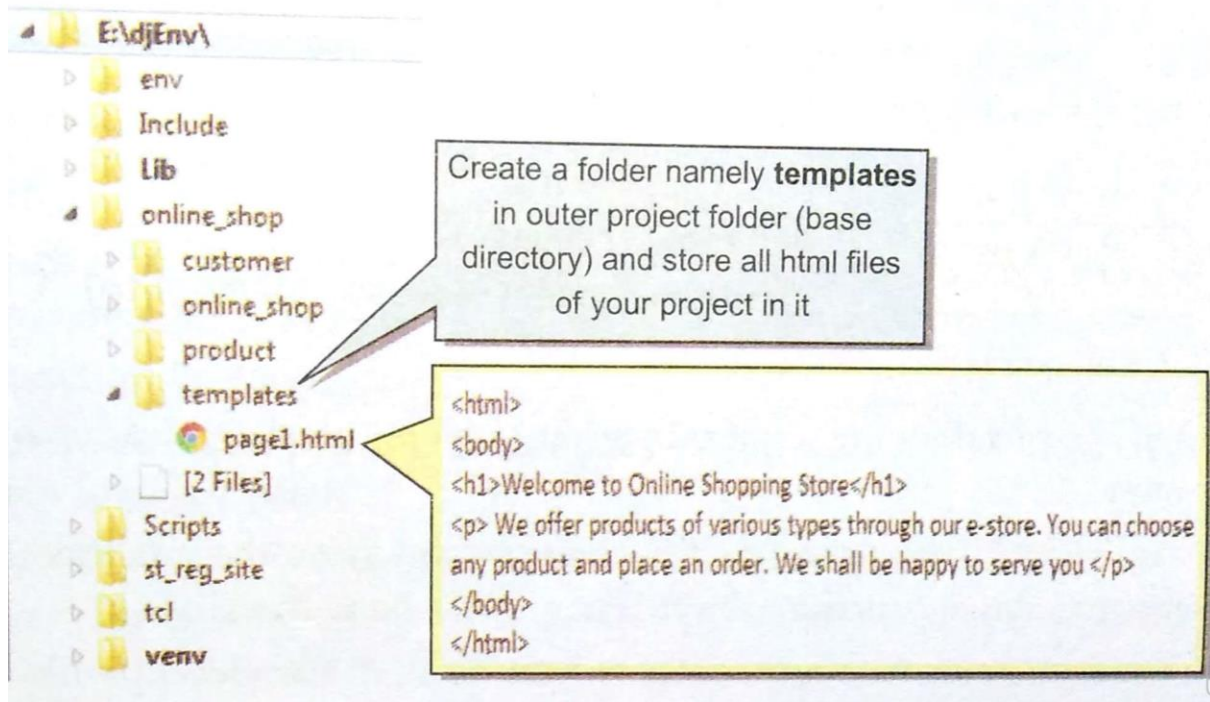
- E.g. if **product** is an *app* under *project Online_shop*, then Model for **product app** will be like:

```
class Product(models.Model) :  
    prod_no = models.IntegerField( )  
    prod_name = models.CharField(max_length = 50)  
    prod_price = models.FloatField( )  
    prod_qty = models.IntegerField( )
```

2. Creating Templates

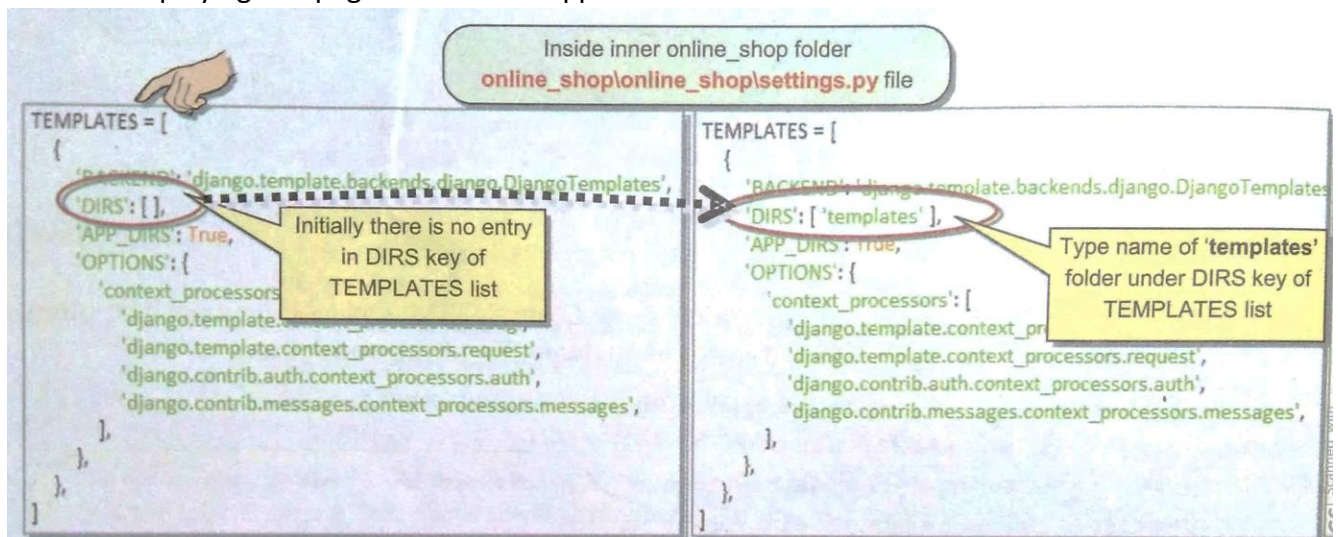
- Templates is the presentation mechanism of Django.
- Steps to create template:
 1. In the BASE DIRECTORY (outer project folder), create a folder by the name **templates**.

2. Next store the desired html files(which are to be displayed in response to a url) in this **templates** folder.



3. Next you need to open the **settings.py** file of your project(inner project name folder) and add the name of templates folder(in string form , i.e. , in quotation marks) in the DIRS setting under TEMPLATES.(see below)

The DIRS key of TEMPLATES setting specifies the place where the server will look for templates while displaying webpages of the web application.



3. Creating Views

- Views receive the request in form of URL and provide response in form of templates which are then presented on the web browser.
- To create views for an app, do the following:
 - (i) The **views.py** file of your app(in app directory), you need to **write view functions** – *one view function* for each html page you want to render. The *view function* is defined in the following format:

```
def <viewname> (request) :  
    return render(request, <htmlfilenamestring>)
```

The view function always takes a request argument

e.g. , we want to display our file **page1.html** which is inside templates folder of base project directory through a view namely **first**, then we shall define a view function as follows inside the **views.py** file of **product** app:

product/views.py file

```
def first(request):  
    return render(request, 'page1.html')
```

Complete name of this view is views.first as this is defined inside views.py file

```
from django.shortcuts import render  
def first(request):  
    return render(request, 'page1.html')
```

4. Creating URL confs

1. Once your model, template and views (MTV) are ready, you need to link all these with URLs through URL Confs. The process of linking URLs and displaying actual templates is also called **URL routing**.

2. Steps:

(i) Import **views.py** file of your apps where you created the view functions for your apps, e.g. we created a view by the name **views.first** in product app's **views.py**. So we shall write following command to import views from *product* app in addition to already existing code.

```
from product import views
```

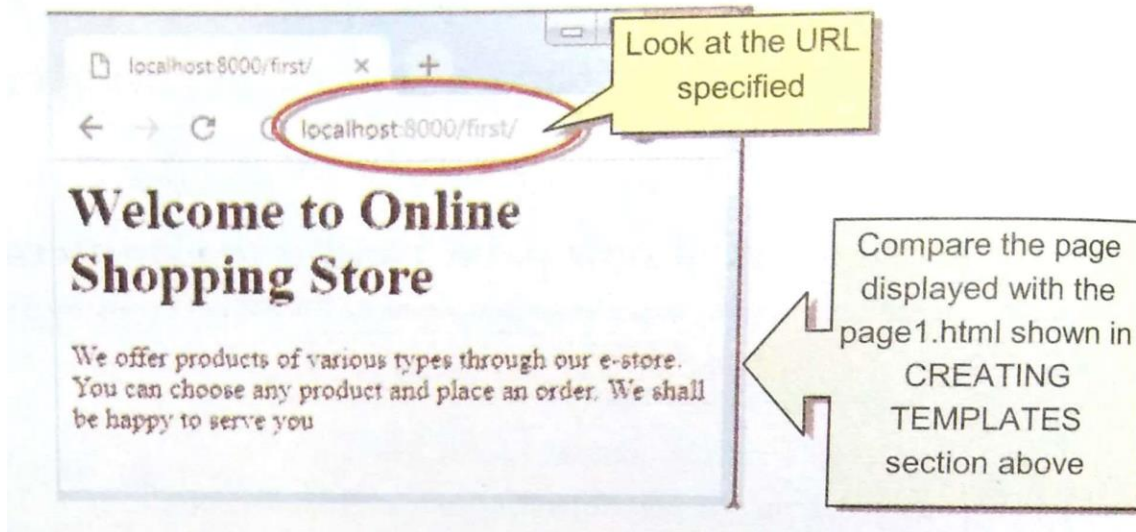
(ii) To define **URL Conf** for your defined views, you need to add following line to **urlpatterns** list in the **urls.py** file.

```
path('first/' , views.first)
```

```
from django.contrib import admin  
from django.urls import path  
from product import views  
  
urlpatterns = [  
    path('admin/' , admin.site.urls),  
    path('first/' , views.first),  
]
```

5. Running Web Application on Server

- (i) Start/activate virtual environment.
- (ii) Go in the BASE DIRECTORY folder of your project.
cd online_shop
- (iii) Now run the Django built-in server for your web application:
python manage.py runserver
- (iv) Open web browser(e.g., start Chrome or Safari on your computer). Open first page of your web application by giving URL as **localhost: 8000** in web browser's address bar.
- (v) Now type the following URL in the address bar of your web browser:
localhost:8000/first/



Writing Dictionary Data to CSV and Text Files

- To write to a CSV file, you need to import python's csv module:
import csv
- For writing data, you need to open a file with .CSV extension in the write ("w") or append ("a") mode.
- Say, you have data to be written in the form of a dictionary called **dict1**, then you need to write the following code:

```
Opens file 'data.csv' in append mode  
with open('data.csv', 'a') as csvfile :  
    csvfile is the name of file-object which will be used inside the with open file construct  
    (Notice all file handling statements within with open file construct are indented inside with open file construct.)  
    The wrt links the fileobject with csv module's writer() method  
    wrt = csv.writer(csvfile)  
    Using wrt now we can write to csv files using writerow() method  
    for key, value in dict1.items():  
        wrt.writerow([key, value])  
    This for loop writes [key, value] from dictionary dict1 to csv file linked to wrt
```

- To write a dictionary's data in a text file, simply open a binary file in **write/append** mode and write into this file.

Questions

- Q1. The architecture of Django consist of?
(a) Models (b) Views (c) Templates (d) All of the above
- Q2. Which option do you use with django-admin to create project in Django?
(a) Newproject (b) myproject (c) createproject (d) startproject
- Q3. Write Django command to create a Django project namely **easysell**.
- Q4. What is the Django command used with **Python** command to start a new app named 'list' in an existing project?
(a) manage.py –newapp list
(b) manage.py newapp list
(c) manage.py -startapp list
(d) manage.py startapp list
- Q5. What is the purpose of settings.py?
(a) To configure settings for the Django project.
(b) To configure settings for an app
(c) To set the data and time on the server.
(d) To sync the database schema.
- Q6. What is the default port used by built in webserver of Django?
(a) 800 (b) 8000 (c) 8800 (d) 127.0.0.1:8000
- Q7. What is the default URL of your django project when you run it on builtin server?
(a) localhost:8000
(b) localhost
(c) 127.0.0.1
(d) 127.0.0.1:8000
- Q8. What is the name of the administrative task management file in Django projects?
(a) manage.py
(b) urls.py
(c) views.py
(d) models.py
