Django Web Framework

First Django Application-

In the previous video we have seen how to create Django Project.

Now, we are now going to create our first application.

Once we create Django project, we can create any number of applications in that project.

Django applications typically follow the MVT pattern. Models define data structures, views handle user requests and interact with models, and templates present data to the user.

Each application can focus on a specific aspect of your website, like user authentication, blog posts, or online shopping carts. This separation keeps your code organized and reduces the complexity within each app.

How to create Django Application-

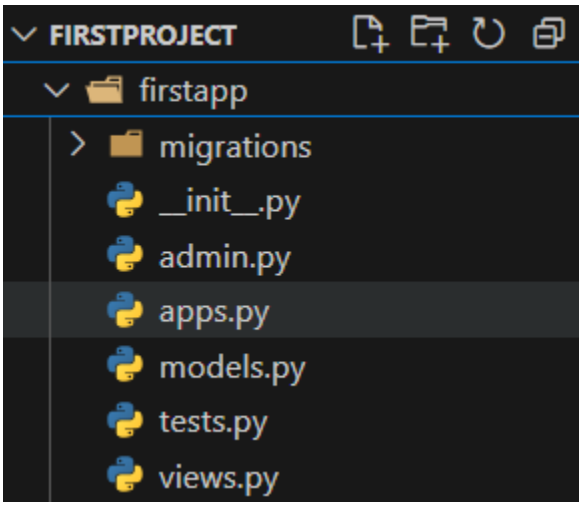
Command to create Django Application-

python manage.py startapp name_of_the_app

```
C:\Users\LENOVO\Documents\djangocourse\firstproject>python manage.py startapp firstapp
```

I have created my first Django application named firstapp inside the project firstproject. Firstproject we had created in the previous video.

A new directory with all these files will be created inside root directory firstproject-



Django takes care of a lot of things by himself. We don't have to explicitly create these files. They are automatically created when we create a Django app.

Let's understand what are all these files-

1.__init__.py-

It is a blank Python script. Because of this special name,Python treated this folder as a package.

2.admin.py-

This file lets you register your app's models with the Django admin interface. If you want to manage your app's data through the admin interface, you'll need to define registrations here.

3.app.py-

This file contains the app configuration class for your Django application. It serves as the entry point for your app and provides settings specific to your app.

4. migrations/ (directory)-

This directory remains empty initially, but it becomes crucial when you define models for your app. Django uses migrations to track changes to your data models and keep your database schema in sync with your code. As you modify your models, Django generates migration files that you'll need to apply to update your database.

5.models.py-

This file is where you define the data models for your application. Models represent the data structures used by your app, such as a User model for user accounts, or a Product model for an online store.

6.tests.py-

This file is intended for writing unit tests for your app's code. Unit tests help ensure that your app's functionalities work as expected and catch regressions when you introduce changes.

7.views.py-

This file is where you define the logic for handling user requests (views) specific to your application. Each view function typically handles a specific URL and interacts with models and templates to generate the appropriate response.

Additional files: As you develop your app further, you might create additional files like:

- **serializers.py:** Mainly used when working with DRF. Used for serializing and deserializing data in various formats like JSON or XML, often used for APIs.
- **forms.py:** Defines forms for user input, allowing users to interact with your app by submitting data through forms.

Note: The most important commonly used files in every project are views.py and models.py.

After creating Application, steps to create and run our first view request and response-

Activity 1: Displaying Hello World!

1.Firstly we have to add the application in settings.py file INSTALLED_APPS list-

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'firstapp',
]
```

2. Create a view for our application in views.py file-

This file is where you define the logic for handling user requests (views) specific to your application.

There are 2 types of views.

1) Function Based Views

2) Class Based Views

In this application we are using Function based views.

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.

def display(request):
    s="<h1>Hello World!</h1>"
    return HttpResponse(s)
```

or we can write directly like this-

```
def display(request):
    return HttpResponse("<h1>Hello World!</h1>")
```

View can accept request as input and perform required operations and provide proper response to the end user.

3.Define URL pattern for our view in urls.py-

This url-pattern will be used by end-user to send request for our views.

The 'urlpatterns' list routes URLs to views.

For functional views we have to do the following 2 activities:

1) Add an import: **from firstapp import views**

2) Add a URL to urlpatterns: **path('display', views.display),**

```python
from django.contrib import admin
from django.urls import path
from firstapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('display/',views.display),
]
```

Now whenever user sends request with urlpattern: display then the display() function view with be executed and provide required response.

4.Start the Django server and send the request-

In the browser type-

http://127.0.0.1:8000/display/

In command prompt(cmd) you will see this-

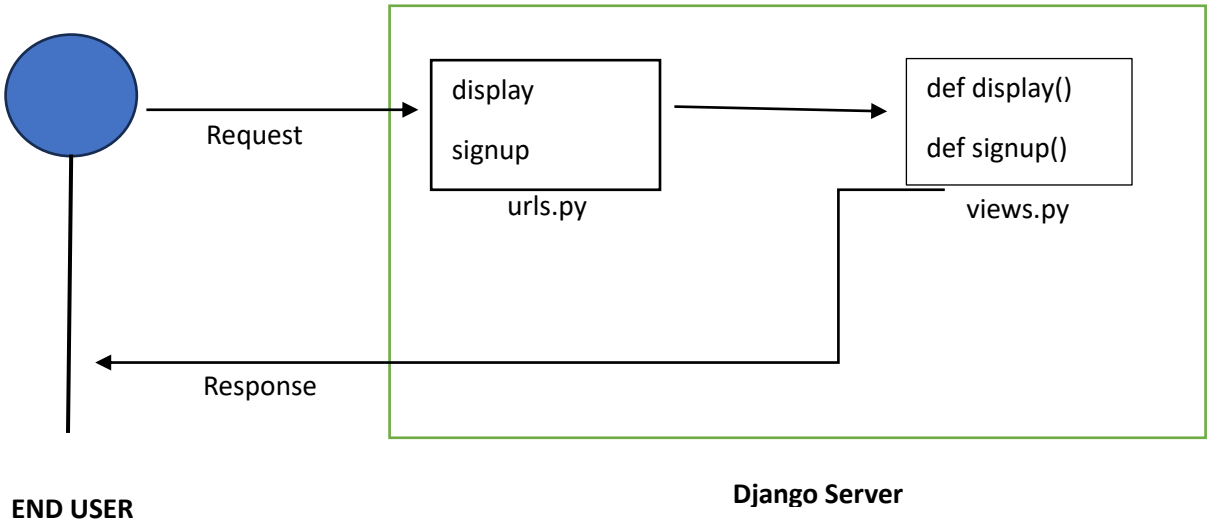
```
[12/May/2024 17:33:44] "GET /display HTTP/1.1" 301 0
[12/May/2024 17:33:44] "GET /display/ HTTP/1.1" 200 21
```

The 200 response means that the display/ url is successfully found.

If a URL or a page is not found then the http response will be 404.

NOTE: We will learn about http response code and http methods later when learning about Django Rest Framework and APIs.

How the HTTP request flow works-



1. Whenever end user sends the request from web browser first Django backend development server will get that request.

2. From the Request Django will identify URL-pattern and by using urls.py, the corresponding or associated view will be identified.

3. The request will be forwarded to the view. The corresponding view function will be executed and the response will be given to the end user as per the view defined.

In short summary what we have done to create the above app

Sequence of Activities related to Django Project- (Steps)

1. Creation of Django project- Django-admin startproject firstproject
2. Creation of Application- python manage.py startapp firstapp
3. Add this application(firstapp) in settings.py INSTALLED_APPS
4. Define views function inside views.py to handle request and response
5. define url-patterns in urls.py
6. Run the development server- python manage.py runserver
7. Open a web browser and send request to the URL.

HOMEWORK EXERCISE-

1. Create a project and application to print your name as response. Follow the above steps.

How to change Django server port-

By default the Django development server runs on port number :8000.

We can also change the port number if a requirement arises.

Command to change port number-

python manage.py runserver 7777

Now server will run on port number: 7777.

If you are following the course series and learning the concepts easily, please subscribe to our YouTube channel. This will motivate us to create more educational and course videos and study material.

Join our growing community of tech enthusiasts! Subscribe to our YouTube channel, where we simplify programming languages in the easiest way possible. Gain a deeper understanding with our clear explanations and receive exclusive notes to enhance your learning journey. Don't miss out on valuable insights – hit that subscribe button now and embark on a programming adventure with us!

Subscribe to our channel:
https://www.youtube.com/@HTEASYLEARNING