

Introduction to Django

1. Django is a Python web framework that enables rapid development of secure and maintainable websites.
2. Follows the model–template–views architectural pattern.
3. Created in the year 2003
4. It's free and open source.
5. It is maintained by the Django Software Foundation (DSF), an independent organization established in the US.
6. Visit <https://www.djangoproject.com/> for more information

Top Backend Technologies

Node.js - is an open-source, cross-platform JavaScript framework that is used to build server-side and networking applications.

Django - is an open-source framework based on Python. It is a web framework from the server's side. Django follows the Model Template View (MTV) architecture.

Spring Boot - is an open-source web framework based on Java that allows developers to build production-grade and standalone applications.

ASP.NET - is an open source web framework, created by Microsoft, for building modern web apps and services that run on macOS, Linux, Windows, and Docker.

Laravel - is a PHP framework that provides a built-in user interface, flexibility, API support, creativity, and an extensive range of various libraries that help in the development process of secure web applications

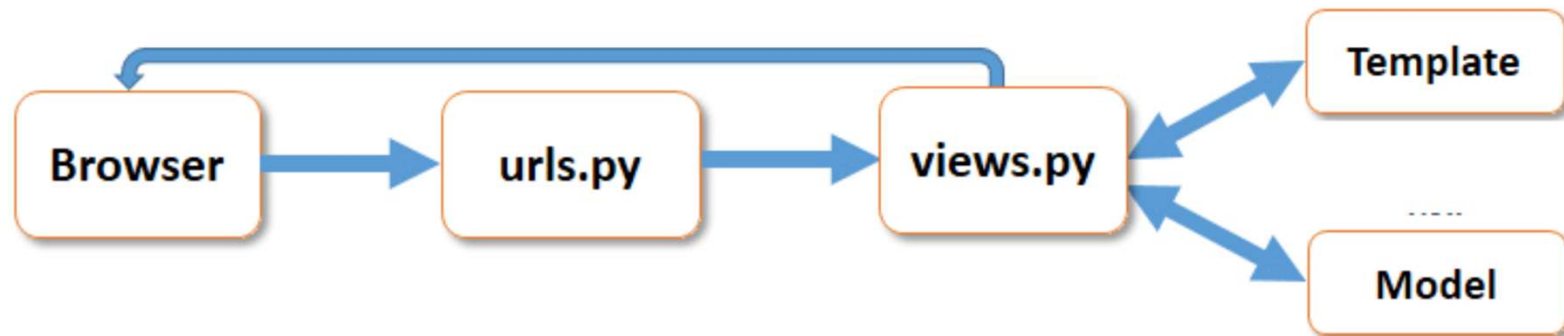
The Course Prerequisites

1. Knowledge of Python Programming
2. Understanding of HTML, CSS and JavaScript
3. Basic idea of any database

Django Framework

Model – View – Template

(Data – Business Logic – Display Page)



Install Python and VS Code

1. Goto <https://www.python.org/downloads/> and download python
2. While installing python select checkbox to add path
3. Goto <https://code.visualstudio.com/download> and download Visual Studio Code
4. Install and then open visual Studio code
5. Click extension icon on the left and then install python (from Microsoft)
6. Close visual studio

Setup Virtual Environment

Search and open powershell as administrator and run following command
Set-ExecutionPolicy RemoteSigned
then press A

Create a folder myproj on desktop
open visual studio code and then open folder myproj from desktop
Goto view menu > click terminal

In terminal type,

```
pip install virtualenv
```

```
python -m venv myenv
```

```
./myenv/scripts/activate (for windows users)
```

```
source ./myenv/bin/activate (for mac users)
```

Type deactivate in terminal to come out of virtual environment

Install django and create project

pip install django

pip list (verify that django has been installed)

django-admin startproject myproj .

python manage.py startapp core

mkdir templates

python manage.py migrate (observe sqlite file on the left)

Create base html page in templates

Base html

```
{% block content %}
```

```
{% endblock %}
```

.....

Go to preferences > settings > emmet > include language > add item –
Django-html : html

Create home,contact,about html in templates

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

```
{% block content %}
```

html content here

```
{% endblock %}
```

Add functions in views.py

```
from django.shortcuts import render
def home(request):
    return render(request,"home.html")
def services(request):
    return render(request,"services.html")
def faq(request):
    return render(request,"faq.html")
def contact(request):
    return render(request,"contact.html")
```

Update url.py

```
from django.contrib import admin
from django.urls import path
from core import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home),
    path('services/', views.services),
    path('faq/', views.faq),
    path('contact/', views.contact),
]
```

Open the website

In terminal type `python manage.py runserver`

Open browser and type `http://localhost:8000/`

Section 1

Develop Todo Application

Setup Virtual Environment

Search and open powershell as administrator and run following command
`Set-ExecutionPolicy RemoteSigned`
then press A

Create a folder myproj on desktop
open visual studio code and then open folder myproj from desktop
Goto view menu > click terminal

In terminal type,

```
pip install virtualenv
```

```
python -m venv myenv
```

```
./myenv/scripts/activate (for windows users)
```

```
source ./myenv/bin/activate (for mac users)
```

Install django and create project

pip install django

pip list (verify that django has been installed)

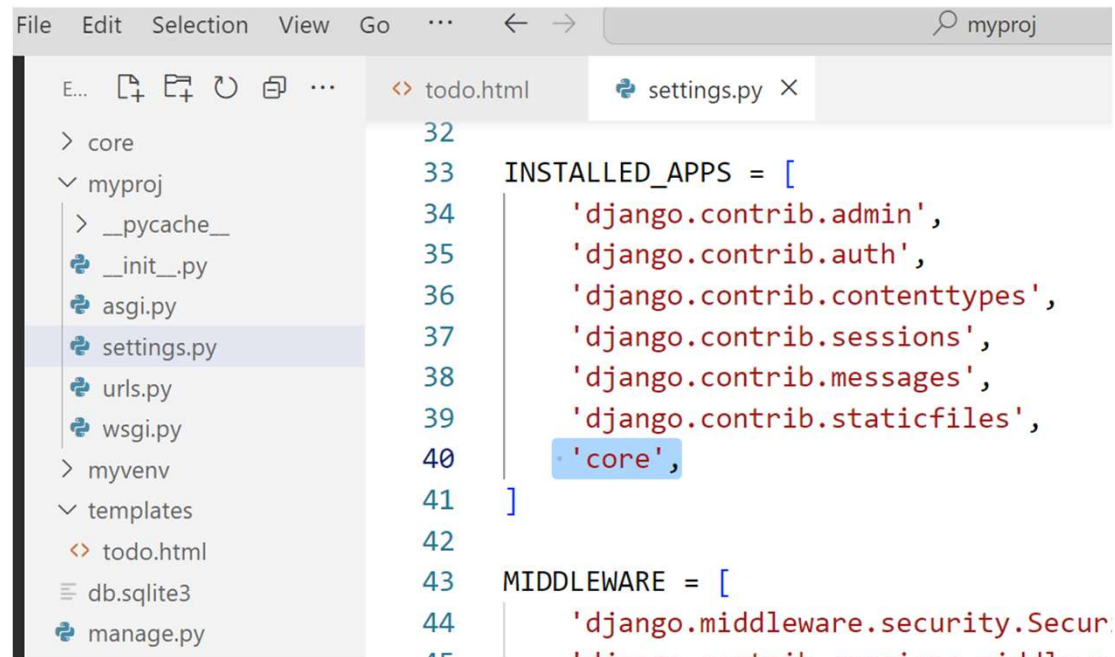
django-admin startproject myproj .

python manage.py startapp core

mkdir templates

python manage.py migrate (observe sqlite file on the left)

Configure core in settings.py

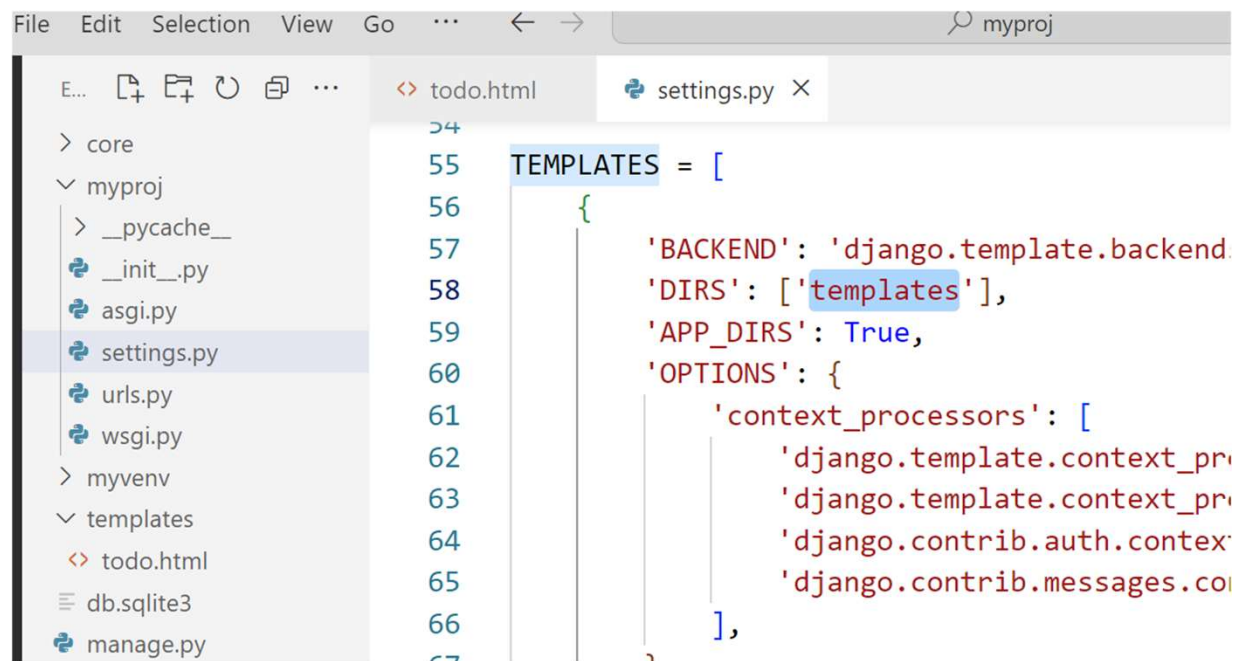


```
File Edit Selection View Go ... < > myproj

> core
✓ myproj
  > __pycache__
  __init__.py
  asgi.py
  settings.py
  urls.py
  wsgi.py
> myenv
✓ templates
  <> todo.html
  db.sqlite3
  manage.py

32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'core',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50 ]
```


Configure templates in settings.py



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'myproj' directory containing 'settings.py', 'urls.py', 'wsgi.py', and 'manage.py'. The 'settings.py' file is selected. The code editor shows the 'TEMPLATES' configuration in 'settings.py'.

```
File Edit Selection View Go ... < > myproj
E... [Icons] ... <> todo.html settings.py X
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': ['templates'],
59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.login_required',
65                 'django.contrib.messages.context_processors.messages',
66             ],
67         },
68     ],
69 ]
```

Create todo model

Update models.py

```
from django.db import models  
class todo(models.Model):  
    task=models.CharField(max_length=50)
```

Run following in terminal:

```
python manage.py makemigrations  
python manage.py migrate
```

Create todo.html inside templates

```
<!DOCTYPE html>
<html lang="en">
  <head><title>Document</title></head>
  <body>
    <h1>Todo Application</h1>
    <form method="POST">
      {% csrf_token %} Cross-Site Request Forgery
      <p><input type="text" name="task" /><button>Add</button></p>
    </form>
    {% for row in todos%}
      <div>
        {{row.task}} <a href="/deleteTodo/?id={{row.id}}">Delete</a>
      </div>
    {% endfor %}
  </body>
</html>
```

Add todo function in views.py

```
from core import models

def todo(request):
    if request.method=="POST":
        task = request.POST.get("task")
        c = models.todo(task=task)
        c.save()
    mytodo = models.todo.objects.all()
    data = {'todos':mytodo}
    return render(request,"todo.html",data)
```

Add deleteTodo function in views.py

```
def deleteTodo(request):  
    if request.method=="GET":  
        todold = int(request.GET.get("id"))  
        models.todo.objects.filter(id=todold).delete()  
    return todo(request)
```

Add path in urls.py

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

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

Run server and view todo in browser

In terminal type > `python manage.py runserver`

open any browser and type following url to view todo app

<http://127.0.0.1:8000/>

Section 2

Setup Admin Interface

Creating Super Admin

In terminal, type > `python manage.py createsuperuser`

Then follow these instructions

username – admin

Email - admin@gmail.com

password – admin

confirm password - admin

bypass validation – press y

Register todo model in admin.py

```
from django.contrib import admin
from core.models import todo
class todoadmin(admin.ModelAdmin):
    list_display=('task',)
admin.site.register(todo,todoadmin)
```

.....

Check for todo in admin site –
python manage.py runserver
<http://127.0.0.1:8000/admin/>

Section 3

Implement Authentication

Setup login page

Create login.html inside template/registration folder

```
<div style="padding:10px">
  <h1>Login Page</h1>
  <form method="post">
    {% csrf_token %}
    {{form.as_p}}
    <button type="submit">Login</button>
  </form>
  <p>New User <a href="/signup/">Register</a> Here</p>
</div>
```

Setup logout

todo.html

```
<a href="/logout">Logout</a>
```

url.py

```
path('logout/',views.logout_view),
```

views.py

```
from django.contrib.auth import logout
```

```
def logout_view(request):
```

```
    logout(request)
```

```
    return todo(request)
```

Setup login redirect & url

Add in settings.py

```
LOGIN_REDIRECT_URL="/"
LOGIN_URL = "/accounts/login/"
```

Add in urls.py

```
from django.urls import path,include
path('accounts/', include("django.contrib.auth.urls")),
```

Add login/logout link in todo.html

```
{% if user.is_authenticated %}  
    {{ user.username }}  
    <a href="/logout">Logout</a>  
{% else %}  
    <a href="/accounts/login">Login</a>  
{% endif %}
```

Setup signup page

create signup.html inside template and signup view

```
<div>
  <h1>Signup Page</h1>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Register</button>
  </form>
</div>
```


Setup signup view

```
from django.shortcuts import redirect
from django.contrib.auth.forms import UserCreationForm
def signup(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('/accounts/login')
    else:
        form = UserCreationForm()
    context = {'form': form}
    return render(request, 'signup.html', context)
```

Setup signup in urls.py

```
from django.contrib import admin
from django.urls import path,include
from core import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.todo),
    path('accounts/', include("django.contrib.auth.urls")),
    path('signup/', views.signup),
]
```

Protect todo view

```
from django.contrib.auth.decorators import login_required
@login_required
def todos(request):
    .....
    .....
```

Section 4

Create

Django Rest API

Setting Up The REST API Project

Create folder djangoapi on desktop and create virtual environment myprojenv

Activate myprojenv

pip install Django

pip install djangorestframework

django-admin startproject djangoapi .

python manage.py startapp api

add following apps in installed apps - settings.py

'rest_framework',

'api',

python manage.py migrate

Api > views.py

```
from rest_framework.response import Response  
from rest_framework.decorators import api_view
```

```
@api_view()  
def getData(request):  
    return Response({"message": "Hello world!"})
```

djangoapi > url.py

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

<http://localhost:8000/> > Click to access getData api

Section 5

Setup Token

Authentication

djangoapi > settings.py

```
INSTALLED_APPS = [  
    'rest_framework.authtoken',  
]  
REST_FRAMEWORK = {  
    'DEFAULT_AUTHENTICATION_CLASSES': [  
        'rest_framework.authentication.TokenAuthentication',  
    ],  
}
```

```
.....  
in terminal type > python manage.py migrate
```

Api > views.py

```
from rest_framework.response import Response
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated
```

```
@api_view()
```

```
@permission_classes((IsAuthenticated,))
```

```
def getData(request):
```

```
    return Response({"message": "Hello world!"})
```

Get api token

Update url.py

```
from rest_framework.authtoken.views import obtain_auth_token  
path('api-token-auth/', obtain_auth_token, name='api_token_auth'),
```

Using postman/thunderclient

Post request, `http://127.0.0.1:8000/api-token-auth/`

body > { "username":"praveen", "password":"Testing123"}

Copy the Token

.....

Get request - **Postman OAuth 2.0 - prefix Token**

Use postman or thunderclient to access api > method :get and headers >

Authorization : Token 1814b600709d835fe5afda569b2ad00848733c0c