

**Django asoslari.**

**Django** – bu bepul va opensource bo’lgan, python dasturlash tili asosida qurilgan, hamda **MTV** *(Model, Template, Views)* arxitekturaviy ko’rinishga asoslangan **framework** hisoblanadi.

**Framework** – bu biron dasturlash tili asosida qurilgan bo’lib, loyiha yaratishda yordam berish, kodlarni standartlashtirish, kodlarni qayta takrorlamaslik uchun ma’lum qoliblarga solib foydalanish kabi vazifalarni bajaradi. Framework dasturchi uchun tayyor qolib va yordamchi.

***Nima uchun Django?***

* **Tez** (G’oyadan dastur shakliga olib o’tish bir qadam)
* **Fullstack** (Ya’ni Web dasturlarni 0 dan boshlab to’liqligicha bitta frameworkda yaratish imkoniyati)
* **Ko’p qirrali** (Django yordamida istalgan turdagi va murakkablikdagi Web dasturlarni yaratish mumkin. Django boshqa web frameworklar va texnologiyalar bilan muammosiz ishlaydi)
* **Xavfsiz** (Django Web dastur foydalanuvchilarning xavfsizligini ta’minlash uchun barcha kerakli vositalar bilan qurollangan.
* **Oson kengayuvchi** (Xoh kichik xoh katta bo’lsin, Django asosida yaratilgan dasturlar osonlik bilan millionlab foydalanuvchilarga xizmat ko’rsata oladi)
* **Barcha turdagi serverlarda ishlaydi**
* **Xizmat ko’rsatish oson**

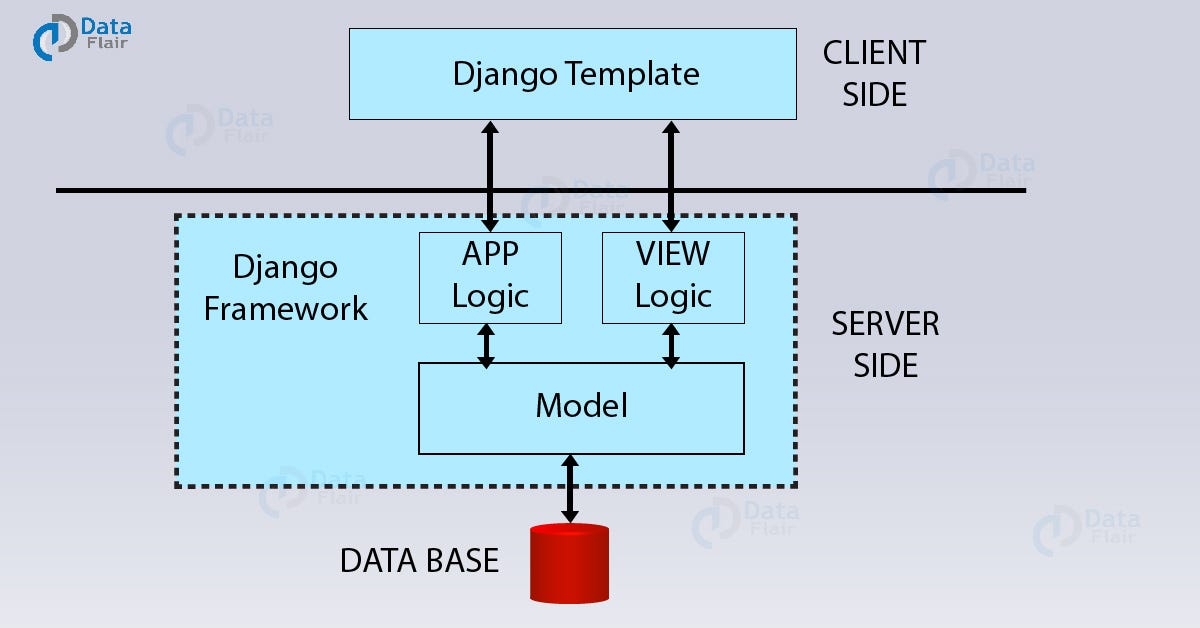
**Terminal bilan tanishish.**

**Terminal** bilan ishlash ko’nikmasi har bir dasturchi uchun muhim narsa hisoblanadi. Terminal orqali biz istalgan ishni buyruqlar ketma-ketligi asosida bajarishimiz mumkin.

**Asosiy buyruqlar:**

* **pwq** – biz hozir qaysi papka ichida turganligimizni ko’rsatadi. (Joriy papkani)
* **dir** – joriy papka ichidagi barcha papkalar ro’yxatini ko’rish.
* **cd Desktop** – joriy papka ichidagi biron papkaga o’tish.
* **cd ..** – joriy papkadan orqaga qaytish.
* **cd Desktop/projects** – orqali papkalar ketma-ketligiga kirish mumkin.
* **ls** – joriy folder ichiadgi papkalar va fayllar ro’yxatini ko’rsatadi.
* **mkdir papka\_nomi** – papka hosil qilish, yangi papka yaratish.
* **rmdir papka\_nomi** – papkani o’chirish.
* **touch faylnomi.keng** – fayl hosil qilish.

**Django arxitekturasi va ishlash tamoyili.**

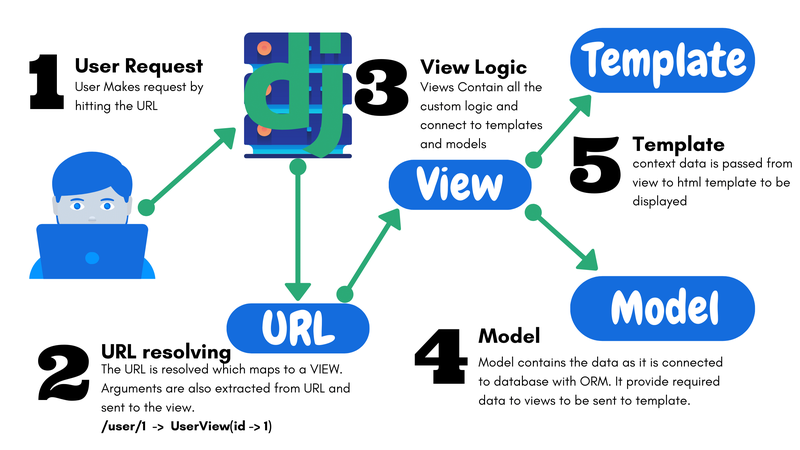


Rasmda django arxitekturasi bitta rasmda tasvirlangan.

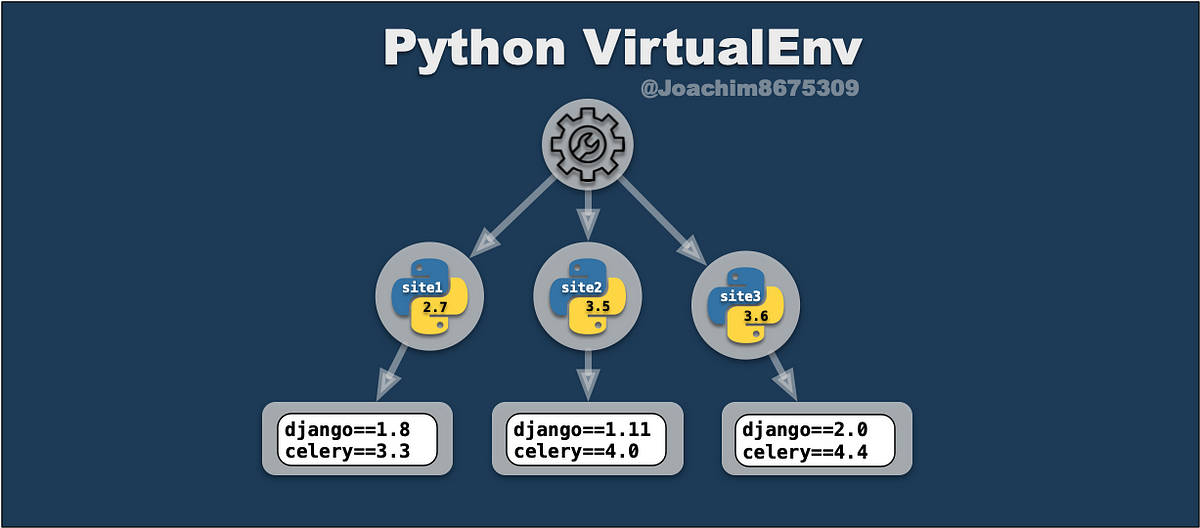
**App Logic** – djangoda loyiharning har bir qismi app’larga bo’linadi. Loyihadagi muhim bir vazifa app ko’rinishida loyihaga bog’lanadi. Bitta loyihada ko’plab **app**’lar bo’lishi mumkin.

**Django Architecture:**

* **M** stands for Model
* **V** stands for View
* **T** stands for Template



**Virtual muhitlar bilan tanishish. Pipenv o’rnatish va sozlash.**



**Virtual muhit** — Django (yoki har qanday Python loyihasi) uchun **mustahkam, xavfsiz va boshqariladigan muhit** yaratadi. Bu — har bir professional dasturchining odatdagi ish jarayonining bir qismidir.

Har bir Django loyihasiga kerakli paketlar, kutubxonalar va versiyalar boshqalardan ajratilgan bo‘ladi.

**Masalan:**  
Agar sizda ikki loyiha bo‘lsa — biri Django 3.2, ikkinchisi esa Django 4.2 talab qilsa, virtual muhit ularning to‘qnashuvini oldini oladi.

Virtual muhitning foydali tomonlari:

1. Izolyatsiya (Ajratilgan muhit)
2. Loyihani boshqarishni osonlashtiradi
3. Global tizimga zarar bermaydi
4. Tizimni toza saqlash

**Virtualenv vs Pipenv** – bu ikkalasi virtual muhitning eng keng tarqalgan turlari.

**pipenv** – virtual muhitini o’rnatish va ishlatish juda ham sodda va qulay. Bu virtual muhit orqali biron yangi modul yoki boshqa narsa o’rnatganimizda packagelar ro’yxati avtomatik ravishda yangilanib boradi.

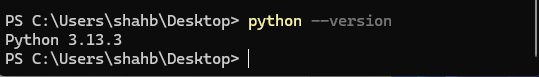
! **virtualenv –** orqali biron yangi package yoki modul o’rnatilganda `freeze > requirements.txt` orqali modullarni yangilash kerak bo’lib qoladi.

Virtual muhitni o’rnatish:

O’rnatishdan avval **python** tizimga o’rnatilganmi yo’qmi tekshirish kerak.

**python --version**

orqali tekshirib olamiz.



**pip install pipenv -** orqali o’rnatish mumkin.

**pipenv --version** –pipenv’ni versiyasini ko’rish o’rnatilgandan keyin.

**pipenv shell** – virtual muhitni yaratish. (ishga tushurish)

**pipenv install django** – orqali djangoni o’rnatish mumkin.

va loyihada chiqishda **deactivate** yoki **exit** qilish kerak.

**Blog loyihasi.**

Loyiha papkasida turgan holda:

1. Virtual muhitni o’rnatamiz:

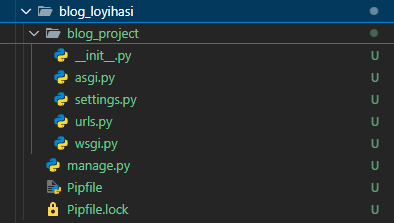
**pipenv shell**

1. Django’ni o’rnatamiz:

**pipenv install Django=4.0**

1. Djangoda loyiha hosil qilamiz:

**django-admin startproject blog\_project .**

****

**Django qismlari bilan tanishish.**

* **Pipfile**
* **Pipfile.lock**

Bu fayllar virtual muhit fayllari. Ular pip o‘rniga **Pipenv** vositasidan foydalanganingizda yaratiladi.

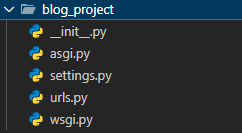
**Pipfile** – bu **loyihangizdagi paketlar ro'yxatini** saqlovchi fayl. U **requirements.txt** ga o‘xshash, lekin undan ko‘ra qulayroq va zamonaviyroq formatda yoziladi.

* **manage.py**

barcha Django buyruqlarini saqlaydi, bu faylni tahrirlashga shoshilmaslik kerak.

Djangoda **default** holatda ma’lumotlar ba’zasi **db.sqlite3** da joylashadi. Bu ma’lumotlar ba’zasi fayli. Bu ma’lumotlar ba’zasini turini loyiha davomida boshqa turga o’zgartirish imkoniyati bor.

Loyihaning core fayllari:



* **\_\_init\_\_.py** – tizimga bu Django loyiha ekanligini bildirib turadi.
* **asgi.py** va **wsgi.py** bu Django loyihasining server interfeysi fayllari bo’lib, ular tashqi web serverlar bilan Django o’rtasida aloqa o’rnatish uchun kerak bo’ladi.
* **settings.py** – loyiha sozlamalarini boshqarish uchun kerak bo’ladi. Django loyihasining *yuragi* hisoblanadi.
* **urls.py** - bu **saytga kelgan so‘rovlarni to‘g‘ri view'ga yo‘naltiradigan** fayl. Django'da har bir URL manzili aynan qaysi funksiya yoki sahifani chaqirishini shu fayl orqali belgilanadi.

Dars nomi: Queryset va model manager

Sekund: 14:40

Loyiha superuser: admin

Parol: admin

# Model manager nima?

Har bitta modelda o’zining default model manageri bor. Model manager bu yordamchi buyruqlar to’plami. Misol uchun **News** va **Category** nomi modellarimizga misol uchun **News.objects.all()** buyrug’idagi ***objects*** bu model manager buyrug’u hisoblanadi.

Djangoda o’zimizga kerakli bo’lgan ***modelmanager*** yaratishimiz mumkin: Buning uchun app papkasini ichida ***managers.py*** nomli fayl yaratib shu faylga kerakli buruqlar yordamida o’zimiz uchun zarur bo’lgan ***modelmanagerni*** yaratib olamiz:

from django.db import models

from .models import News

class PublishedManager(models.Manager):

  def get\_queryset(self):

    return super().get\_queryset().filter(status=News.status.Published)

**models.py** Ichida chaqirib ishlatib ko’ramiz:

models ga chaqirib ishlata olmadik, shu sabab o’zini Ichida chaqirdik:

from django.db import models

from django.utils import timezone

# Create your models here.

class PublishedManager(models.Manager):

  def get\_queryset(self):

    return super().get\_queryset().filter(status=News.Status.Published)

class Category(models.Model):

  name = models.CharField(max\_length=150)

  def \_\_str\_\_(self):

    return self.name

class News(models.Model):

  class Status(models.TextChoices):

    Draft = "DF", "Draft"

    Published = "PB", "Published"

  # id = models.IntegerField(primary\_key=True, unique=True)

  title = models.CharField(max\_length=250)

  slug = models.SlugField(max\_length=250)

  body = models.TextField()

  image = models.ImageField(upload\_to='news/images') #Rasmlar bilan ishlaganda "pillow" kutubxonasini o'rnatish kerak bo'ladi.

  # pipenv install Pillow

  category = models.ForeignKey(Category, on\_delete=models.CASCADE)

  publish\_time = models.DateTimeField(default=timezone.now)

  created\_time = models.DateTimeField(auto\_now\_add=True)

  updated\_time = models.DateTimeField(auto\_now=True)

  status = models.CharField(max\_length=2, choices=Status.choices, default=Status.Draft)

  objects = models.Manager() #default django manager

  published = PublishedManager()

  class Meta:

    ordering = ["-publish\_time"]

  def \_\_str\_\_(self):

    return self.title

**python manage.py shell** - QuerySet orqali published manager buyrug’idan foydalanib published qilingan newslarni chiqarib olish:

**from news\_app.models import News, Category**

News.**published**.all()

# Loyihaga **Static** fayllarni bog’lash

Settings.py faylida:

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [BASE\_DIR / 'static']

STATIC\_ROOT = BASE\_DIR / 'staticfiles'

STATICFILES\_FINDERS = [

  'django.contrib.staticfiles.finders.FileSystemFinder',

  'django.contrib.staticfiles.finders.AppDirectoriesFinder',

]

MEDIA\_URL = 'media/'

MEDIA\_ROOT = BASE\_DIR / 'media/'

Urls.py faylida:

if settings.DEBUG:

    urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

    urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Djangoda context processor bor va u base.html kabi view orqali ma’lumot uzatib bo’lmaydigan template’larga ma’lumot uzatishda foydalaniladi. App ichida ***context\_processor.py*** faylini yaratib olamiz. (Fayl nomi ixtiyoriy bo’lishi mumkin.)

Quyidagicha kodlarni yozdik faylimizga:

from .models import News, Category

def info():

  latest\_news = News.published.all().order\_by("-publish\_time")[:10]

  category = Category.objects.all()

  context = {

    'latest\_news': latest\_news,

    'category': category

  }

  return context

Endi esa ***./config/settings.py*** loyiha sozlamalariga o’zgartirish kiritib olamiz:

TEMPLATES = [

    {

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

        'DIRS': [BASE\_DIR / 'templates'],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

                'news\_app.context\_processor.info'

            ],

        },

    },

]

Qo’shilgan qism: ***AppNomi.ContextProcessorFaylNomi.YaratilganFunksiyaNomi***

Endi istalgan template orqali bu **context processor** yuboraotgan ma’lumotlarni qabul qilib olishimiz mumkin.

**base.html** faylida chaqirib ko’ramiz:

Navbar qismida chaqirib ishlatib ko’rdik:

<!-- Nav Start -->

<div class="classynav">

  <ul>

    <li><a href="#">Home</a></li>

    <li>

      <a href="{% url 'catagory' %}">Category</a>

      <ul class="dropdown">

        {% for category\_name in category %}

        <li><a href="index.html">{{ category\_name.name }}</a></li>

        {% endfor %}

      </ul>

    </li>

    <li><a href="#">About Us</a></li>

    <li><a href="{% url 'contact' %}">Contact Us</a></li>

  </ul>

</div>

<!-- Nav End -->