**DJANGO**

Masaüstünde klasör yarat : **blog**

Open cmd in there (Klasör adres çubuğuna gel ve cmd yaz orada açar cmd yi)

**C:\Users\Demir\Desktop\blog>** **pip install virtualenv**

**C:\Users\Demir\Desktop\blog>** **virtualenv venv** //venv isimli bir klasör yarattı

**C:\Users\Demir\Desktop\blog>** **venv\Scripts\activate** // aktif ettik sanal zımbırtıyı

**(venv) C:\Users\Demir\Desktop\blog>** //En başta env görüyorsan olmuştur.

**(venv) C:\Users\Demir\Desktop\blog>deactivate //** yazarak kapatıyoruz sanal zımbırtıyı.

* Sanal python ortamını kurmamdaki amaç ana python dosyamdan ayırmam projemi. Bir izalasyon yaratmam.

**C:\Users\Demir\Desktop\blog>** **venv\Scripts\activate** // aktif ettik sanal zımbırtıyı

**(venv) C:\Users\Demir\Desktop\blog> pip install django**

**(venv) C:\Users\Demir\Desktop\blog> django-admin startproject blog .** // proje yaratıldı.

* Şimdi pycharmda blog adlı projemizi açalım ve settings en altta dil te zaman Europe/Istanbul yapalım
* dJango bize sunucu desteğide sağlıyor.

**(venv) C:\Users\Demir\Desktop\blog> dir** // baktık manage dosyası burada mı diye.

04/04/2020 22:25 <DIR> .idea

04/04/2020 22:25 <DIR> blog

04/04/2020 22:08 645 manage.py

**(venv) C:\Users\Demir\Desktop\blog>python manage.py runserver** // Serverı çalıştırdık

… Django version 3.0.5, using settings 'blog.settings'

Starting development server at <http://127.0.0.1:8000/> …

* <http://127.0.0.1:8000/> ADRESİNİ TARAYICINA GİR VE MUHTEŞEM SONUCU GÖR 😊
* Komut satırından **Ctrl+c** ile server sonlandırılabilir.

**(venv) C:\Users\Demir\Desktop\blog>python manage.py runserver 8080**

* 8080 Port numarasından erişilebilirsin.
* … Starting development server at <http://127.0.0.1:8080/> …

**SUPERUSER CREATE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**> venv\Scripts\activate //** Daha önce yaratıtğımız venv sanal projesini aktif ettik.

**> python manage.py runserver**  // çalıştırdık.

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. //dedi

\* Ctrl + c ile çıkıp. '

**> python manage.py migrate** // dedik. Ve admin sayfası vs. aktif dilmiş oldu.

**(venv) C:\Users\Demir\Desktop\blog> venv\Scripts\activate**

**(venv) C:\Users\Demir\Desktop\blog> python manage.py runserver**

* <http://127.0.0.1:8000/admin> yazıp admin paneline girdik.
* Bu arada komut satıtımızda server açık ama biz başka işlemlerde yapacağız sanal makinada o yüzden yeni bir cmd daha açıp orada işlerimizi yapıyoruz. Örneğin bir sanal user yaratacağız.

**(venv) C:\Users\Demir\Desktop\blog> python manage.py createsuperuser**

Kullanıcı adı (leave blank to use 'demir'): yunus

E-posta adresi:

Password:

* Siteden gir bak yetkilerine.
* Gruplar/yetkiler//öğrenci/öğretmen vs
* Kişi ekle sil güncelle vs.

**Uygulama yaratma\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**(venv) C:\Users\Demir\Desktop\blog> python manage.py startapp post**

**Post/models** içine

class Post(models.Model):  
 title = models.CharField(max\_length=120) # Max\_lengt belirtilmek zorundaymış.  
 text = models.TextField()  
 published\_date = models.DateTimeField()

**blog/settings/installedapps** altına bildiriyoruz yazdığımız modeli belirtiyoruz

‘post’,

**(venv) C:\Users\Demir\Desktop\blog> python manage.py makemigrations** // ve migrate altına yeni bir dosya oluştu artık migrate edebiliriz. Sql’e çevirdi yazdıkalarımızı

* Şimdi bakacağız.

**(venv) C:\Users\Demir\Desktop\blog>python manage.py sqlmigrate post 0001**

BEGIN; --CREATE TABLE "post\_post" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "title" varchar(120) NOT NULL, "text" text NOT NULL, "published\_date" datetime NOT NULL);

COMMIT;

Post/admin

from .models import Post  
#from post.models import Post  
  
admin.site.register(Post)

**Admin panelini özelleştirmek\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**> venv\Scripts\activate**

**> python manage.py runserver**

* <http://127.0.0.1:8000/admin>
* yunus 123

class PostAdmin(admin.ModelAdmin):  
  
 list\_display = ["title", "published\_date"] # Ekranda neler görünsün.  
 list\_display\_links = ["title", "published\_date"]  
 list\_filter = ["published\_date"]  
 search\_fields = ["title", "text"]  
 # list\_editable = ["title"]  
  
 class Meta:  
 model = Post  
  
admin.site.register(Post, PostAdmin)

**Views**

**Post view’e aşağıdaki kodu yazıyoruz. Yani bu demekki bu viewe gelinince hoşgeldiniz yazısı dönülsün.**

from django.shortcuts import render,HttpResponse  
  
def home\_view(request):  
 return HttpResponse('<h>Hoşgeldiniz</h>')

**Daha sonra bu viewi blog/urls e tanıtıyoruz. Hangi url yazılınca bu gözüksün diye.**

from post.views import home\_view

path('', home\_view)

kök dizine tıklyınca bu metod gelsin diyoruz. Yani

>venv\Scripts\activate

>python manage.py runserver

Starting development server at <http://127.0.0.1:8000/>

Artık buraya girdiğinde hoşgeldiniz yazısı gözükmüş oluyor.

**Urls:**

**>python manage.py startapp home**

Home isimli bir uygulama oluşturduk. Ve viewine yukarıdaki view’e yaptığımızı yaptık.

**Post dediğimiz de bir ana sayfa olcak onun alt sayfaları olacak bunların tutulacağı urls dosyası lazım bu otomatikman gelmedi bunu elimizle oluşturyoruz. Ve olası alt url’leri oluşturuyoruz.**

from django.urls import path  
urlpatterns = [  
 path('index', post\_index),  
 path('details', post\_details),  
…  
]

**Şimdi bunların view’lerini oluşturacağız.**

from django.shortcuts import render,HttpResponse  
  
def post\_index(request):  
 return HttpResponse('<h>İndex Page</h>')  
  
def post\_detail(request):  
 return HttpResponse('<h>Detail Page</h>')

…

**Tekrar post/urls gidip**

from .views import \*

**blog/urls e post/urls i tanıt.**

from django.urls import path, include

path('post/', include('post.urls') ),

**Template:**

* Blog/settings/templats metodu içine az sonra oluşturacağımız templates dosyasının yolunu gösteriyoruz. BASE\_DIR içinde.

TEMPLATES = [  
 {…  
 'DIRS': [os.path.join(BASE\_DIR, "templates")],

…

* Blog / sağ click new directory (template). Template sağ click new htlm (home)
* Home /views dizinine gidip

def home\_view(request):  
 return render(request, 'home.html', {})

Dinamik hale getirmek:

* Home/ views
* def home\_view(request):  
   if request.user.is\_authenticated:  
   context = {'isim' : 'yunus'}  
   else:  
   context = {'isim': 'Misafir'}  
   return render(request, 'home.html', context)
* burada context değişkeniyle karşı tarafa bir değişken göndermiş olduk karşı taraftan ise.

Templates/home {{isim}} diyerek karşıladık.

# **Django Shell ve Database API**

1. **Open shell**

**(myvenv) ~/djangogirls$ python manage.py shell**

* Şu an Django'nun etkileşimli konsolundayız. Önce tüm post'larımızı görüntülemeyi deneyelim. Bunu aşağıdaki komut ile yapabiliriz:

1. **In [3]: from post.models import Post**
2. **In [4]: Post.objects.all()**

**Out[4]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>]>**

**In [5]: from django.utils import timezone**

**In [6]: suan = timezone.now()**

1. **Create:**

**In [8]: Post.objects.create(title = "Deneme123", text = "deneme123", published\_date = suan)**

**Out[8]: <Post: Deneme123>**

**In [9]: Post.objects.all()**

**Out[9]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>, <Post: Deneme123>]>**

1. **Create 2:**

**In [10]: nesne = Post()**

**In [11]: nesne.title = "Deneme1"**

**In [12]: nesne.text = "Deneme1"**

**In [13]: nesne.published\_date = "suan"**

**In [14]: Post.objects.all()**

**Out[14]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>, <Post: Deneme123>]>** //KAYDETMEMİŞ KAYDETMEK İÇİN SAVE DİYEEECEĞİZ.

**In [18]: nesne.save()**

**In [19]: Post.objects.all()**

**Out[19]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>, <Post: Deneme123>, <Post: Deneme1>]>**

**In [20]: nesne.title = "Guncellenen baslik"**

**In [21]: nesne.save()**

**In [22]: Post.objects.all()**

**Out[22]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>, <Post: Deneme123>, <Post: Guncellenen baslik>]>**

1. **Delete:**

**In [23]: a = Post.objects.get(id=3)**

**In [24]: a**

**Out[24]: <Post: Deneme123>**

**In [25]: a.delete()**

**Out[25]: (1, {'post.Post': 1})**

**In [27]: Post.objects.all()**

**Out[27]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>, <Post: Guncellenen baslik>]>**

1. **Filter:**

**In [30]: Post.objects.filter(title\_\_contains = "sg")** // büyük küçük duyarlı ama sqlite zaten

değil.

**Out[30]: <QuerySet [<Post: sg>]>**

**In [30]: Post.objects.filter(title\_\_icontains = "sg")** // büyük küçük suyarsız getir.

**In [34]: Post.objects.filter(published\_date\_\_lte = suan )** //least than equal şuan

**Out[34]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>, <Post: Guncellenen baslik>, <Post: Deneme123>]>**

**In [39]: Post.objects.order\_by('published\_date')**

**Out[39]: <QuerySet [<Post: sg>, <Post: sdfsdfdfsfsfs>, <Post: Guncellenen baslik>, <Post: Deneme123>]>** exit()

**İndex and Details:**

**1.Statik**

**Post/views.py**

from .models import Post

def post\_index(request):  
 posts = Post.objects.all()  
 return render(request, 'post/index.html', {'posts': posts})

**temaplate/post/index.html**

<body>  
{% for post in posts %}  
 {{ post.id }}<br>  
 {{ post.title }}<br>  
 {{ post.text }}<br>  
 {{ post.published\_date }}<br>  
{% endfor %}  
  
</body>

**Post/view/detail**

from django.shortcuts import render,HttpResponse, get\_object\_or\_404

def post\_detail(request):  
 post = get\_object\_or\_404(Post, id=4)  
 context = {'post' : post}  
 return render(request, 'post/detail.html', context)

* get\_object\_or\_404 : Sayfayı bulamazsa bize 404 arızası versin diye bunu import ettik. Ama DEBUG = True olduğu için yani debug modunda olduğumuzdan tam da kullanıcının alacağı gibi almıyoruz arızayı. Tüm kod bitince Debug = False yapılır ve kod canlıya alınır.

**2.Yarı Dinamik**

**Post/urls**

path('<int:id>', post\_detail),

**post/views**

def post\_detail(request, id):  
 post = get\_object\_or\_404(Post, id=id)  
 context = {  
 'post' : post  
 }  
 return render(request, 'post/detail.html', context)

**Artık site urlsinden** <http://127.0.0.1:8000/post/1> veya 2 ve ya 3 diyebiliriz.

**3. Dinamik**

**1. post/index içindeyiz. Başlığa tıklanılınca ilgili nesneye ait** post.get\_absolute\_url metodu var ona git.

<a href="{{ post.get\_absolute\_url }}"> {{ post.title }} </a><br>

1. **post/model içine gidiyoruz. Yukarıda gidilen metod. Bu metod kendi id sini döndürüyor. Nereye urls içine adı post olan urllerden detaile id isminde kendi id sini döner.**

from django.urls import reverse

def get\_absolute\_url(self):  
 return reverse('post:detail', kwargs={'id': self.id})

1. post/urls buraya gelen id alınır ve yerleştirilir.

path('<int:id>', post\_detail, name ="detail"),

1. Burasıda ilgili id nin detail sayfasına gider.

{{ post.id }}<br>  
{{ post.title }}<br>  
{{ post.text }}<br>  
{{ post.published\_date }}<br>

**GET- POST Nedir ne değildir?**

HTTP: HYPER TEXT TRANSFER PROTOCOL: Sunucu ve kullanıcıların birbiriyle haberleşme metodlarıdır.

GET: Sunucudan veri almak için (html) / very göndermek için de kullanılır ama kullanıcı giriş işlemleri vs yapılmaz. Ve sınırlıdır. Adres çubuğunda açık olarak görülür.

POST: Güvenli ve sınırsız very gönderimi yapılır.

[14/Apr/2020 13:13:25] "GET /admin HTTP/1.1" 301 0 // sayfalar yükleniyor get ile alıyor zımbırtıları

[14/Apr/2020 13:13:31] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 1929

[14/Apr/2020 13:13:31] "GET /static/admin/css/login.css HTTP/1.1" 200 1233

…

[14/Apr/2020 13:13:36] **"POST /admin/login/?next=/admin/ HTTP/1.1" 302 0 //Kullınıcı girişi: Post**

[14/Apr/2020 13:13:36] "GET /admin/ HTTP/1.1" 200 5445

[14/Apr/2020 13:13:36] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 412

…

[**http://127.0.0.1:8000/admin/post/post/?q=deneme**](http://127.0.0.1:8000/admin/post/post/?q=deneme)

[14/Apr/2020 13:30:05] "GET /admin/post/post/?q=deneme HTTP/1.1" 200 5170

[14/Apr/2020 13:30:05] "GET /admin/jsi18n/ HTTP/1.1" 200 7337

* ara: deneme yazdık. ?q= deneme dedi arama işlemini get ile yaptı ve adres çubuğunda görüldü. ?q = query demek

**[14/Apr/2020 13:28:53] "POST /admin/post/post/6/change/ HTTP/1.1" 302 0** // Formu güncelledim.

[14/Apr/2020 13:28:53] "GET /admin/post/post/ HTTP/1.1" 200 7425

[14/Apr/2020 13:28:53] "GET /admin/jsi18n/ HTTP/1.1" 200 7337

**Özet:** Sayfa istekleri ve aramalar GET ile, Kullanıcı giriş, Db ve form işlemleri ise POST metodu ile yapılır.

VİDEO 19 CREATE POST GET BAKARSIN NOT İÇİN KODDA DA AÇIKLAMALAR YER ALMAKTADIR.

20- UPDATE

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

$ git status

[...]

$ git add --all .

$ git status

[...]

$ git commit -m "Veritabanından gönderileri göstermek için template'ler değiştirildi."

[...]

$ git push

* [PythonAnywhere](https://www.pythonanywhere.com/consoles/)'e bağlanalım ve **Bash konsolu**'na gidelim (veya yeni bir konsol açalım) ve şunu çalıştıralım:

PythonAnywhere komut-satırı

$ cd $USER.pythonanywhere.com

$ git pull

[...]

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***