**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 Por numarasından erişilebilir yapabilirsin.

…

Starting development server at <http://127.0.0.1:8080/>

…

------------------------------ders 6

>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.

>(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 sananl makina 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()

yazdık.

Post/settings/installedapps altına bildiriyoruz yazdığımız modeli belirtiyoruz

‘post’,

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

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:

**Django nedir?**

Django, (framework). Bir web iskeleti, websitesi geliştirmeyi hızlandıran ve kolaylaştıran bir grup bileşendir. So it is The library of libraries

Bir web sitesi geliştirdiğinizde benzer türde bileşenlere her zaman ihtiyacınız olur: kullanıcılar için kimlik denetimi (üye olma, üye girişi, üye çıkışı), websiteniz için bir yönetim paneli, formlar, dosyaları yüklemek için bir yol, vs.

Django'nun **url çözücüsü** (urlresolver) tarafından yapılıyor (websitesi adresine URL - Uniform Resource Locator - deniyor, dolayısıyla url çözücü ismi mantıklı oluyor). Çok akıllı değildir - kalıpların bir listesini alır ve URL'yi eşleştirmeye çalışır. Django kalıpları üstten alta doğru denetler ve eşleşen bir şey varsa isteği ilgili işleve (view olarak adlandırılan) aktarır.

Tüm ilginç şeyler view (görünüm) fonksiyonunda yapılır:

Öyleyse bir **virtual environment** (diğer adıyla virtualenv) kuralım. Virtualenv Python/Django kurulumunuzu her proje için ayrı tutup izole eder. Bu, bir websitesine yapacağınız değişikliklerin diğer geliştirdiklerinize yansımayacağı anlamına gelir. Muazzam, değil mi?

manage.py site yönetimine yardımcı olan bir komut dosyasıdır. Bu dosya sayesinde, başka herhangi bir şey kurmadan bilgisayarımızda bir web sunucusunu başlatabileceğiz.

settings.py dosyası, web sitesinizin ayarlarını içerir.

Bir mektubu nereye götüreceğini kontrol eden postacının hakkında konuştuğumuzu hatırlıyor musun? urls.py dosyası urlresolver(urlçözümleyici) tarafından kullanılan kalıpların bir listesini içerir.

**Web sunucusunu başlatmak**

manage.py dosyasının bulunduğu dizinde olmalıyız (djangogirls klasörü). Konsol üzerinden python manage.py runserver komutunu çalıştırarak web sunucusunu başlatabilirsiniz:

--

Peki bir nesne nedir? Nesne bir özellikler ve hareketler bütünüdür. Eğer bir kediyi modellemek istiyorsak, Kedi nesnesini oluştururuz ve bu nesne şöyle özelliklere sahip olur: renk, yas, ruh\_hali (örneğin; iyi, kötü, uykulu ;)) Ayrıca Kedi bazı hareketlere sahiptir: miyavla, tirmala ya da beslen (bu durumda kediye biraz KediMamasi vermemiz gerekir ki o da kendine ait özellikleri olan başka bir nesne olarak tanımlanabilir, bu durumda onun özellikleri arasında da tat yer alabilir). Yani aslında ana fikir, gerçek nesneleri kod içinde özellikleri (nesne özellikleri) ve hareketleri (metodlar) ile tanımlamak.

Öyleyse blog gönderilerini nasıl modelleyeceğiz? Bir blog tasarlamak istiyoruz, degil mi?

Cevaplamamız gereken soru: Bir blog gönderisi nedir? Özellikleri ne olmalıdır?

Tabii ki blog gönderimizin içeriği için yazı ve bir de başlık lazım, değil mi? Kimin yazdığını da bilsek iyi olur - dolayısı ile bir de yazara ihtiyacımız var. Son olarak, gönderinin ne zaman yaratıldığını ve yayınlandığını da bilmek isteyebiliriz.

Post

--------

title

text

author

created\_date

published\_date

Bir blog gönderisi ile ne tür şeyler yapılabilir? Gönderiyi yayınlayan bir method olması güzel olurdu, değil mi?

Bu yüzden publish metoduna ihtiyacımız olacak.

Django'da modeller özel bir çeşit nesnedir - veritabanı'na kaydedilir. Veritabanında, kullanıcılar, blog gönderileri, vs. ile ilgili bilgileri saklarız. Verilerimizi depolamak için SQLite veritabanını kullanacağız. Veritabanındaki bir modeli, sütunları (alan adı) ve satırları (veri) olan bir hesap çizelgesi olarak düşünebilirsiniz**.**

### Uygulama oluşturma

Her şeyi derli toplu tutmak için, projemizin içinde ayrı bir uygulama oluşturacağız. Her şeyin en başından düzenli olması çok iyidir. Bir uygulama oluşturmak için aşağıdaki komutu konsolda çalıştırmamız gerekiyor (djangogirls dizininden manage.py dosyasının bulunduğu yer):

(myvenv) C:\Users\Name\djangogirls> python manage.py startapp blog

Uygulamamızı oluşturduktan sonra, Django'ya bunu kullanmasını söylememiz lazım. Bunu mysite/settings.py dosyasından yapacağız -- kod editörümüzde açalım. INSTALLED\_APPS'ı bulup ] karakterinin üzerindeki satıra 'blog', yazmamız lazım. Sonuç aşağıdaki gibi görünmeli:

mysite/settings.py

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'blog',

]

str'nin her iki tarafında da 2 tane alt çizgi (\_) kullandığımızdan emin olalım. İki alt çizgi Python dilinde sık kullanılır, bazen "dunder" olarak da anılır ("double-underscore"un kısaltması).

* models.CharField - belirli bir uzunluktaki metinleri tanımlamak için kullanılır.
* models.TextField - bu da uzun metinleri tanımlar. Blog gönderileri için biçilmiş kaftan, değil mi?
* models.DateTimeField - bu da gün ve saati tanımlamada kullanılır.
* models.ForeignKey - başka bir modele referans tanımlar.

### Modeller için veritabanında tablo oluşturma

Son adımımız yeni modelimizin veritabanına eklenmesini sağlamak. Öncelikle Django'nun modelimizde bazı değişiklikler yaptığımızı bilmesini sağlamalıyız. (Daha yeni oluşturduk!) Komut satırına gidelim ve python manage.py makemigrations blog yazalım. Şöyle görünmeli:

**Not:** Düzenlediğiniz dosyaları kaydetmeyi unutmayın. Aksi takdirde, bilgisayarınız önceki sürümü çalıştırarak beklenmedik hatalar verebilir.

Django bize veritabanımıza uygulayabileceğimiz bir taşıma (migration) dosyası oluşturdu. python manage.py migrate blog yazdığın zaman şunu görmelisin: