Arkadaşlar merhaba,

Bu videomuzda Django’da komutlarını anlatacağım.

-------------- commands --------------

Django built-in komutlara sahip olup, bunlardan bir kaçını daha önceki videolarda görmüştük. Komutları

**python manage.py <komut>**

şeklinde çalıştırıyoruz. Neler yaptığımızı hatırlayalım runserver, migrate, makemigrations, startapp, startproject, createsuperuser. Yeri geldiğinde başka komutlarda kullanacağız.

Tüm komutların listesine;

**python manage.py help**

Komutu ile ulaşabilirsiniz. Komutların detaylarına ise

**python manage.py help <komut>**

yazarak ulaşabilirsiniz. Ben bu konuda sizlere built-in komutları anlatmayacağım. Kendi komutlarımızı nasıl oluşturabileceğimizden bahsedeceğim.

Komutlar uygulamalarımız içerisinde management/commands/ klasörü içerisinde bulunur. Öncelikle klasörümüzü oluşturalım ve \_\_init\_\_.py mizi yükleyelim.

Şimdi komutu yazacağımız dosyayı oluşturabiliriz. (sample.py). Dosyamızı açalım ve Command classımızı yazmaya başlayabiliriz. Her komut Command sınıfına yazılır ve Command sınıfı BaseCommand dan türetilmelidir.  
BaseCommand import edelim ve ekrana ‘Command Çalıştı’ yazdıralım.

Help değişkenine yardım metni atanır. Command için yazılan kodlar handle() metodu içerisinde yazılır.

from django.core.management.base import BaseCommand   
  
class Command(BaseCommand):

help = 'Komut hakkında yardım bilgisi'

def handle(self**,** \*args**,** \*\*kwargs):  
 print('Command Çalıştı')

Komutu çalıştırmak için :

**python manage.py sample**

yazdığımızda. Ekrana ‘Command Çalıştı’ yazdığını görüyoruz.

Komutlar ile birlikte argümanlarda gönderebiliriz. Command sınıfında argüman eklemek için add\_arguments() metodu kullanılır.

Django argümanlar için Python’un argparse kütüphanesini kullanır. İki çeşit argüman vardır zorunlu ve isteğe bağlı argümanlar. Argüman başına tire ekleyerek isteğe bağlı yapabiliriz. Şimdi zorunlu ve isteğe bağlı argümanlar yazalım.

Argümanlar parser.add\_argument() metodu ile eklenmektedir.

Öncelikle zorunlu argümanlarımızı oluşturalım. 2 tane zorunlu argümanımız olsun. user\_id argümanı oluşturalım int olsun. yardım metnini yazalım. username argümanı oluşturalım string olsun. yardım metnini yazalım.

def add\_arguments(self**,** parser):  
 # Zorunlu argüman  
 parser.add\_argument('user\_id'**,** type=int**,** help='user\_id yardım bilgisi')  
 parser.add\_argument('username'**,** type=str**,** help='username yardım bilgisi')

3 tane de isteğe bağlı argüman oluşturalım. Prefix argümanımız olsun string olsun. isteğe bağlı yapmak için başına – koymamız gerektiğini söylemiştim. İki tane de koyabiliriz. Admin argümanı oluşturalım eğer girilirse true olarak atansın. Yardım bilgisini ekleyelim. Users argümanı girelim ve int seçelim. Yardım bilgisini ekleyelim. liste olarak gönderebilmek için nargs argümanını + olarak girelim. Kısaltma da ekleyebiliriz. Onun için ilk argüman olarak u girelim. Artık u ve users olarak argüman gönderebiliriz.

# İsteğe Başlı Argüman  
 parser.add\_argument('--prefix'**,** type=str**,** help='prefix yardım bilgisi')  
 parser.add\_argument('--admin'**,** action='store\_true'**,** help='admin yardım bilgisi')  
 parser.add\_argument('-u'**,** '--users'**,** nargs='+'**,** type=int**,** help='users yardım bilgisi')

Argümanlarımızı ekledik şimdi handle() metodunda gelen argümanları yakalayalım.

def handle(self**,** \*args**,** \*\*kwargs):  
  
 user\_id = kwargs['user\_id']  
 product\_id = kwargs['product\_id']  
 prefix = kwargs['prefix']  
 admin = kwargs['admin']  
 users = kwargs['users']  
  
 print('Positional user\_id : ' + str(user\_id))  
 print('Positional product\_id : ' + str(product\_id))  
  
 if prefix:  
 print('Optional prefix : ' + prefix)  
  
 if admin:  
 print('Optional admin selected')  
  
 if users:  
 print('Optional list : ' + str(users))

Şimdi komutumuzu çalıştırabiliriz.

**python manage.py sample 1 2 –prefix emre –admin –users 1 2 3 4 5**

Opsiyonel alanlardan sadece birtanesini yazalım

**python manage.py sample 1 2 –prefix emre**

Artık veritbanımıza çok ekleme çıkarmaya yapmaya başlayacağız. Kendimize bir veri datası hazırlayalım ve bunu istediğimiz zaman veritabanına yükleyebilelim. Bunun commandlar ile yapabiliriz.

**python manage.py populate**

komutunu çalıştırdığımızda hazırladığımız data veritabanına yüklensin.

Comands/ klasörüne populate dosyası oluşturalım.

BaseCommand ve veri ekleyeceğimiz Product, Category, User, UserDetail modellerini de import edelim ve sınfımızı ve handle() metodunu oluşturalım. Herhangi bir argüman göndermeyeceğiz sadece komut çalıştırınca handle() metodunu çalıştıracağız onun için add\_arguments() metodunu oluşturmamıza gerek yok.

Veritabanına verileri daha önce öğrendiğimiz metodlardan olan bulk\_create ike gerçekleştirelim. Hatırlarsanız bulk\_create veritabanında toplu veri eklemede kullanılıyordu. Normalde bizim Product modelimiz save() yapıldığında otomatik slug oluşturmaktadır. Fakat bulk metodları save() metodunu çağırmadan ekleme işlemini gerçekleştirmektedir. Bunun için bizim ayrıca slug vermemiz gerekmektedir.

from django.core.management.base import BaseCommand  
from learning.models import UserDetail**,** Product  
from django.contrib.auth.models import User

def handle(self**,** \*args**,** \*\*kwargs):

UserDetail.objects.delete()  
 User.objects.delete()  
 Product.objects.delete()

User.objects.bulk\_create([  
 ])  
  
 UserDetail.objects.bulk\_create([  
 ])  
  
 Product.objects.bulk\_create([])

Komutumuzu çalıştırarak tablolarımıza verilerimizi yükleyelim.

**python manage.py populate**