Merhaba arkadaşlar,

Bir önceki konumuzda model kavramına giriş yaptık ve model oluşturup genel özeliklerini öğrendik. Şu anda temel bilgiler anlatıyorum. Konular ilerledikçe anlaşılması daha zor olacak onun için temel bilgileri lütfen tekrar edelim.

Bu videomuzda veritabanına nasıl veri ekleyeceğimizi, güncelleyeceğimizi ve listeleyebileceğimizi öğreneceğiz. İlk aşamada veritabanındaki verilere ulaşmak için django shell kullanacağız. Daha sonra benzer işlemleri views dosyalarımızda da yapacağız. Kullanacağımız metodlar aynı olacak.

Django shelll açmak için terminalde:

**python manage.py shell**

yazıp entere basıyoruz.

Öncelike veritabanı ilişki kurabilmek için modeli import etmemiz gerekmektedir. product ve user modelimizi shelle import edelim.

**from learning.models import Product**

**from django.contrib.auth.models import User**

------------------ veritabanına ekleme ve güncelleme ------------------

Veritabanına veri eklemek ve güncellemek için Model sınıfının save() metodunu kullanıyoruz.

Bir adet kullanıcı ekleyelim.

**user = User(username=’editor’, email=’editor@learning.udm’)**

Kullanıcı bilgilerini user nesnesine atamış olduk. Bunları veritabanına işlemek için save() methodunu kullanmalıyız.

**user.save()**

create() methodunu kullanarak da veritabanına ekleme yapabiliriz.

**User.objects.create(username=’personel’, email=’personel@learning.udm’)**

Veritabanından kontrol ettiğimizde eklenmiş olduğunu görüyoruz.

Kullanıcı ismini değiştirelim. Hala user nesnemizde kayıt ettiğimiz kullanıcı bilgileri durmaktadır.

**user.username = ‘yonetici’**

**user.save()**

Bir tanede ürün eklemesi yapalım. Ürünlerde bildiğiniz gibi User modeli ile ilişkikli author alanı bulunmaktadır.

**product = Product(name=’Shell Product’, content=’Shellden yeni ürün ekliyoruz’, author=user)**

Evet şimdi product değişkenine ürün nesnemizi atadık. save() yaparak veritabanına kayıt edebiliriz.

Veritabanındaki verileri güncellemenin diğer bir yolu da update() methodunu kullanmak. 1 numaralı kullanıcının email adresini güncelleyelim. update() methodu geriye güncellenen nesne sayısını döndürür.

**User.objects.filter(id=1).update(email=’update@learning.udm’)**

------------------ veritabanından veri çekme ------------------

Veritabanından veri çekmek için Model Manager (model yöneticisi) ile QuerySet yani sorgu seti oluştururuz. Her modelin en az bir adet Manager’ı vardır ve modelismi.objects olarak ulaşılır.

**Product.objects**

QuerySet nedir?

Querysetler veritabanındaki bir nesne kolleksiyonunu temsil eder. Mesela Ürünler.

QuerySetlerin sıralanabilir., filtrelenebilir. Bu işlemler için Manager metodlarını kullanırız. Şimdi bu metodları inceleyelim.

Tüm nesneleri listelemek için **all()** methodunu kullanırız. Veritanbanındaki kullanıcıları users değişkenine atadık. Şimdi bunları ekrana bastıralım.

**users = User.objects.all()**

**for user in users:**

**print(user)**

Sorgu setinin ilk nesnesini getirmek için **first()** methodunu kullanırız. İlk nesnenin bilgilerini user değişkenine atayalım ve daha sonra elemanlarına ulaşalım.

**user = User.objects.first()**

**user.username**

**user.is\_staff**

Sorgu setinin son nesnesini getirmek için **last()** methodunu kullanırız.

**User.objects.last()**

Eğer kaydı nasıl çekeceğimizi biliyorsak **get()** kullanırız. Sadece bir tane kayıt gelmesi gerekmektedir.

Eğer birden fazla kayıt gelirse [**MultipleObjectsReturned**](https://docs.djangoproject.com/en/2.1/ref/exceptions/#django.core.exceptions.MultipleObjectsReturned) hatası döner.

Eğer kayıt gelmez ise [**DoesNotExist**](https://docs.djangoproject.com/en/2.1/ref/models/instances/#django.db.models.Model.DoesNotExist) hatası verir. Bunun için genellikle unique (benzersiz) alanlar için veya primary\_key ile birlikte kullanılır.

**User.objects.get(id=1)**

Sorgu setindeki kayıt tarihi en son olan nesneyi getirmek için latest() methodu kullanılır

**User.objects.latest(date\_joined)**

Sorgu setindeki kayıt tarihi en önce olan nesneyi getirmek için earliest() methodu kullanılır

**User.objects.earliest(date\_joined)**

Sorgu setinde kayıt olup olmadığını tespit etmek için exists() methodu kullanılır. Eğer kayıt varsa True yoksa False döner.

**User.objects.filter(id=1).exists()**

------------------ veritabanına veriyi çek, eğer veri yoksa ekle ------------------

**get\_or\_create()** metodu ile veritabanından bir veriyi çekmeye çalıştığımızda eğer varsa çekeriz yoksa çekmeye çalıştığımız veriyi veri tabanına ekletebiliriz. get\_or\_create() geriye tupple olarak döner.

obj olarak eklenen veya çekilen nesne bilgisini döndürür

created olarak da eğer eklendiyse True, çekildiyse False döndürür.

defaults kısmında eklenecek olan diğer alanları belirtiriz.

**obj, created = User.objects.get\_or\_create(**

**username='emre',**

**defaults={'email': ‘emre@learning.udm’},**

**)**

------------------ veritabanındaki veriyi güncelle, eğer veri yoksa ekle ------------------

**update\_or\_create()** methodu ile veritabanından bir veriyi güncellemeye çalıştığımızda eğer veri varsa güncellenir yoksa güncellemeye çalıştığımız veri veritabanına ekletebiliriz. update\_or\_create() geriye tupple olarak döner.

obj olarak eklenen veya çekilen nesne bilgisini döndürür

created olarak da eğer eklendiyse True, güncellendiyse False döndürür.

defaults kısmında eklenecek olan diğer alanları belirtiriz.

**obj, created = User.objects.update\_or\_create(**

**username ='Emre',**

**defaults={'email': ‘emre@learning.udm’},**

**)**

------------------ veritabanından toplu veri çekme ------------------

in\_bulk() methodu ile veritabanındaki kayıtları primary key veya benzersiz alan ile eşleştirerek çekebiliriz. dict olarak döner.

User tablosundaki tüm kayıtları getirir.

**User.objects.in\_bulk()**

User tablosundaki primary keyi 1 ve 2 olan kayıtlar

**User.objects.in\_bulk([1,2])**

Blog tablosunda slug alanı filtered\_word olan kayıtları getirir.

**Blog.objects.in\_bulk(['filtered\_word'], field\_name='slug')**

------------------ veritabanına toplu ekleme ------------------

bulk\_create(obj) methodu ile verilen nesne listesi bir tane sorgu çalıştırılarak veritabanına eklenir.

**User.objects.bulk\_create([**

**User(username='Emre'),**

**User(username ='Emre'),**

**], 1)**

------------------ veritabanında toplu güncelleme ------------------

bulk\_update() methodu ile verilen nesne listesi bir tane sorgu çalıştırılarak veritabanına eklenir. Veritabanından User modelindeki tüm nesneleri çekelim. Daha sonra 0 ve 1. Elemanlarının is\_staff bilgisini değiştirelim ve is\_staff bilgisini toplu olarak güncelleyelim.

**objects = User.objects.in\_bulk()**

**objects[0].is\_staff = 1**

**objects[1].is\_staff = 1**

**User.objects.bulk\_update(update.values(), ['is\_staff'])**

------------------ veritabanından veri silme ------------------

Veritabanından veri silmek için delete() methodunu kullanırız. Bu kod veritabanındaki primary\_key 2 olan kaydı siler ve geriye silinen nesne sayısını döndürür.

**user = User.objects.get(pk=2)**

**user.delete()**

Tüm ürünleri silmek için ise bu şekilde bir kod çalıştırabiliriz. Fakat veritabanına yeniden veri eklememiz gerekeceği için çalıştırmıyorum. Siz deneyebilirsiniz.

**products = Product.objects.all()**

**products.delete()**

------------------ veritabanındaki verileri sayma ------------------

count() methodu sorgu setindeki nesne sayısını döndürür.

**User.objects.count()**

veritabanındaki toplam kullanıcı sayısı

**User.objects.filter(is\_staff=1).count()**

veritabanındaki is\_staff=1 olan kullanıcı sayısı

Eğer QuerySet oluşturulmuşsa count() methodunun çalıştırması ekstra performans kaybında neden olur böyle durumlarda len() kullanılması daha uygundur.

**users = User.objects.all()**

**user.count()** # yanlış kullanım

**len(users)** # doğru kullanım

Bu videomuzda veritabanına nasıl veri ekleyeceğimizi, güncelleyeceğimizi ve listeleyebileceğimizi öğrendik.

Bir sonraki videomuzda ise sorgu setlerindeki kayıtların nasıl filtreleneceğini öğreneceğiz.