**Django Signals:**

|  |  |
| --- | --- |
| django.db.models.pre\_init & django.db.models.post\_init | Sent before or after a models’s \_init\_() method is called |
| django.db.models.signals.pre\_save & django.db.models.signals.post\_save | Sent before or after a model’s save() method is called |
| django.db.models.signals.pre\_delete & django.db.models.signals.post\_delete | Sent before or after a models’ delete() method or queryset delete() method is called |
| django.db.models.signals.m2m\_changed | Sent when a ManyToManyField is changed |
| django.core.signals.request\_started & django.core.signals.request\_finished | Sent when an HTTP request is started or finished |

**Django Caching:**

| **Strategy** | **Description** |
| --- | --- |
| Memcached | A memory-based cache server is the fastest and most efficient |
| FileSystem Caching | Values of the cache are stored as separate files in a serialized order |
| Local-memory Caching | This is used as the default cache strategy by Django if you haven’t set anything. It is per-process as well as thread-safe. |
| Database Caching | Cache data will be stored in the database and works very well if you have a fast and well-indexed DB server. |

**Select\_Related ve Prefect\_Related**

Select: Yukarıdaki örnekte olduğu gibi kullanacağımız kesin olan yerlerde objeyi çekerken ilişkili diğer objeyi select\_related ile çekebiliriz.

product = Product.objects.select\_related('category').first()

print(product.name, product.category.name)

Burda ürünü getirirken bağlı olduğu kategoriyide beraberinde çektiğimizden kategorinin fieldlarına erişmeye çalıştığımızda ekstra bir query atmaz.

Prefect: prefetch\_related, many-to-many ve reverse relation ile birbirine bağlı objeleri tek seferde çekmemizi sağlar.

categories = Category.objects.prefetch\_related('products').all()

for category in categories:

print(category.products.all())

Burada kategoriyi çekmek için bir query, kategorilere bağlı ürünleri çekmek için ayrı bir query atar. Daha sonra python tarafında çektiği ürünleri ilgili kategori ile ilişkilendirir. Binlerce kategori, binlerce ürün bile olsa toplamda 2 query atar.

**11. Describe the inheritance styles in Django?**

Django offers three inheritance styles:

1. **Soyut temel sınıflar:** Üst sınıfın her alt model için yazmak istemediğiniz verileri korumasını istediğinizde bu stili kullanırsınız.
2. **Çok tablolu devralma:**Bu stili, varolan bir modelde bir alt sınıf kullanmak istediğinizde ve her modelin kendi veritabanı tablosuna sahip olmasını istediğinizde kullanırsınız.
3. **Proxy modelleri:** Modelin alanını değiştirmeden modellerle Python düzeyinde davranışı değiştirmek için bu stili kullanırsınız.

**36. Django bir talebi nasıl işler?**

Django Server bir istek aldığında, sistem hangi Python kodunun yürütülmesi gerektiğini belirlemek için bir algoritma izler. Algoritmayı özetleyen adımlar şunlardır:

* Django, kök URL yapılandırmasını denetler.
* Ardından, Django istenen URL'nin eşleşmesi için URLconf'taki tüm değişken URL kalıplarına bakar.
* URL eşleşirse, ilişkili görünüm işlevini döndürür.
* Daha sonra, herhangi bir veri gereksinimi için bu uygulamanın Modelinden verileri isteyecek ve tarayıcı tarafından işlenen ilgili Şablona iletecektir.
* Django, URL'lerin hiçbiri istenen URL'yle eşleşmiyorsa bir hata işleme görünümü gönderir.

# F Nesnesi ile Toplu Güncelleme

Bir modelin bir alanının mevcut değerini almak için F nesnesini kullanırız. Böylece, güncelleme yaparken, önce varolan değeri almak için bir sorgu, ardından yeni değeri yazmak için ikinci bir sorgu atmaya gerek kalmaz. Tabii sorgu sayısı güncellemek istenen kayıt sayısına bağlı olarak katlanarak artacaktır.

Örneğin, tüm çalışanların maaşını 500 TL artırmak için aşağıdaki işlemleri yapmam gerekir:

employees = Employee.objects.all()for employee in employees:  
 employee.salary = employee.salary + 500  
 employee.save()

Aynı işlemi F nesnesi kullanarak aşağıdaki gibi tek bir sorguda yapabilirim:

from django.db.models import Femployees = Employee.objects.all()  
employees.update(salary=F('salary') + 500)