**KONU : DJANGO’YA GİRİŞ**

DERS 1: KURS BİLGİSİ (1 SAYFA)

* kurs hakkında bilgiler

DERS 2: DJANGO NEDİR (1 SAYFA)

* django hakkında bilgiler

DERS 3: KURULUM, PROJE VE UYGULAMA OLUŞTURMA ( 3 SAYFA)

* virtualenv ve django kurulum
* proje oluşturma ve dosyaları inceleme
* settings.py
* uygulama oluşturma ve dosyaları inceleme

DERS 4: MODEL OLUŞTURMA, MIGRATION VE YÖNETİCİ PANELİ (3 SAYFA)

* model oluşturma ve temel bilgiler
* migration
* yönetici oluşturma
* uygulamanın yönetici paneline eklenmesi
* yönetici panelini özelleştirme

DERS 5: URL, VIEW VE TEMPLATELER

**KONU : MODEL KATMANI**

DERS 6: MODEL ALAN ÖZELLİKLERİ ve MODEL ALAN TÜRLERİ (4 SAYFA)

* model alan özellikleri
* model alan türleri

DERS 7: DJANGO SHELL – EKLEME, GÜNCELLEME, SİLME, LİSTELEME (shell) (4 SAYFA)

* veritabanına ekleme ve güncelleme
* veritabanından veri çekme
* veritabanından veri silme
* veritabanındaki verileri sayma
* veritabanına veriyi çek, eğer veri yoksa ekle
* veritabanındaki veriyi güncelle, eğer veri yoksa ekle
* veritabanından toplu veri çekme
* veritabanına toplu ekleme
* veritabanında toplu güncelleme

DERS 8: FİLTRELEME, SIRALAMA VE DİĞER METHODLAR (6 SAYFA)

* filter() metodu
* verileri sıralama ve limit
* değer metodları
* karşılaştırma metodları
* tarih metodları

DERS 9: MODEL METODLARI, MODEL YÖNETİCİLERİ ve MODEL KALITIMI (8 SAYFA)

* model metodları
* model yöneticileri
* model kalıtımı

DERS 10: MODEL İLİŞKİLERİ ve KOMUT OLUŞTURMA (9 SAYFA)

* django komutları ve özel komut oluşturma
* bire – bir ilişki
* çoka – çok ilişki
* çoka – bir ilişki
* select\_related ve prefect\_related
* Prefect nesnesi

DERS 11: AGGREGATE(), ANNOTATE(), Q(), F()

* Q nesnesi ile karışık sorgular yapma
* F nesnesi
* aggregate
* annotate

DERS 12: DİĞER MODEL AYARLARI

* Meta ayarları
* Birden çok veritabanı kullanma
* Database transactions
* Optimization

**KONU : VIEW KATMANI**

Validation Konusunda

Models.full\_clean() anlatılmalı

<https://stackoverflow.com/questions/4441539/why-doesnt-djangos-model-save-call-full-clean>

Even though the idea of enforcing validation on Model level seems right, Django does not do this by default for various reasons. Except for some backward-compatibility problems, the authors probably don't want to support this because they fear this could create a false feeling of safety when in fact your data are not guaranteed to be always validated. Some ORM methods (e.g. bulk\_create or update) don't call save() and thus are unable to validate your models. In other words, it is hard to guarantee the validation, thus they've decided not to pretend it.

If you need this for multiple models, you can create a simple mixin that overrides the save() method and calls full\_clean() before super. Do note that this might cause the validation to be run twice in some cases, like when using ModelForm. It might not be that of an issue though if your validation routines are side-effect free and cheap to run.

**KONU : VIEWS KATMANI**

DERS : SIGNALS

DERS : DOSYA İŞLEMLERİ