------------------ kurulum ------------------

Arkadaşlar merhaba,

Bu bölümde Django kurulumu gerçekleştireceğiz ve Django hakkında genel bilgiler vereceğim.

Ben geliştirme ortamı olarak bu eğitim boyunca PAYÇARM, veritabanı olarak ise SQL Lite kullanacağım.

Django kurulumu için bilgisayarınızda python ve pip kurulu olması gerekmektedir. Daha önceki konularda python, pip, PAYÇARM nasıl kurulduğu anlatılmıştı. Şimdi Django kurulumuna başlayabiliriz.

Eğer PAYÇARM Professional Edition kullanıyorsanız direk Django projesi oluşturabilirsiniz. Ben sıfırdan sanal ortam ve Django kurulumunu göstereceğim.

Öncelikle terminali açıyoruz.

**mkdir project** (ile klasör oluşturuyoruz)

**cd project** (oluşturduğumuz klasöre girelim)

**virtualenv stocksproject** (projemiz için yeni sanal ortam yaratalım)

**.\Scripts\activate** Sanal ortamı aktifleştirmek için ilgili klasöre gidiyoruz ve komutunu çalıştırıyoruz.

virtual envoriement ile sanal bir ortam oluşturur ve projelerinizi bu ortam içinde çalıştırmamıza yarar. Bu sanal ortam projeyi dış ortamdaki paket, sürüm değişiklikleri vs gibi problemlerden korur. Aktifleştirildiğinde gördüğünüz gibi başında sanal ortam ismi gelmektedir.

**pip freeze**

komutu ile kurulu paketleri kontrol edelim. Gördüğünüz gibi mevcutta paket bulunmamaktadır.

**pip install Django**

komutu ile Django kurulumu yapabiliriz.

**pip freze**

kurulu paketlere tekrar baktığımızda Django geldiğini görebiliriz.

Eski bir versiyon için

**pip install Django==2.2**

İki tane eşittir işaret ve sonrasına versiyon numarası ile de kurulum yapabiliriz.

**django-admin –version**

yazarak kurulan Django versiyonunu kontrol edebiliriz.

Şimdi Django projemizi oluşturabiliiz.

**django-admin startproject stocks**

Yazıyoruz. Buradaki stoks proje adımız oluyor.

------------------ proje oluşturma ------------------

Evet şimdi oluşturduğumuz projeyi PAYÇARM içine alalım. Pycharm -> Open -> Django Proje mizi seçiyoruz. Burada dikkat edelim Proje klasörünü değil (Project deği) Django Projesini (stocks) seçmemiz gerekmektedir.

Şimdi PAYÇARM için virtualenv ayarlarını yapmamız gerekiyor.Settings -> Project: stocks -> Project Interpreter den ayar simgesine tıklıyoruz. Proje klasörünün içindeki Scripts içerisinden python.exe seçiyoruz. Gördüğünüz üzere kurulu paketlerimiz buraya geldi. Daha önce terminalden Django projemizi oluşturmuştuk. Proje yapısına baktığımızda içerisine bir adet stocks adında klasör ve manage.py dosyası eklendi.

manage.py dosyası terminal ekranından komutlarımızı çalıştırmaya yarayacaktır. Bu dosya ile herhangi bir işlem yapmayacağız.

\_\_init\_\_ py dosyası bulunduğu klasörün bir PAYTON modülü olduğunu göstermektedir.

settings.py Django ayarlarının bulunduğu dosyadır. Bunun içerisinden genel Django ayarlarını yapıyoruz.

urls.py dosyasından browserden istek yapıldığında hangin kodun çalıştırılması gerekiyorsa oraya yönlendirme yapıyoruz.

wsgi.py – sunucumuzun ayarlarının yapıldığı dosyadır.

Şimdi web sunucumuz çalışıyormu kontrol edelim. Bunun için manage.py ile runserver komutunu çalıştırıyoruz.Django otomatik olarak kendini 8000 portunda çalıştırıyor.

**python manage.py runserver**

ctrl + c yaparak sunucuyu durdurabiliriz.

Eğer 8000 portunda başka bir şey çalıştırıyorsak portu elle vererek başka portta da çalıştırabiliriz.

**python manage.py runserver 8081**

Açılan karşılama sayfası gördüğünüz gibi İngilizce. Biraz önce Django ayarlarını settings.py den ayarlayabildiğimizi söylemiştik.

------------------ settings.py ------------------

Şimdi settings.py dosyasını biraz inceleyelim.

Burada **DEBUG = True** görüyoruz.

Bu sistem hataların ekrana basılmasını sağlamaktadır. Bu sayede hatalarımızı görebilir ve gerekli yerleri düzenleyebiliriz. Bu development yani geliştirme süreci bittiğinde False olarak değiştirilmelidir. Aksi taktirde güvenlik açığı oluşturacaktır.

**INSTALLED\_APPS** da hali hazırda kurulan uygulamalar gösterilmektedir. Biz paket indirdiğimizde veya uygulama yazdığımızda

Aktifleştirmek için bu alana eklememiz gerekmektedir.

**MIDDLEWARE**, ara katman anlamına gelmektedir. Ne işe yarar dersek gelen HTTP isteklerini filtrelemeye yarar. Örnek vermek gerekirse;

Tarayıcıdan bir url girmeye çalıştığınızda önce Middleware devreye girer kullanıcının giriş yapıp yapmadığına bakar. Eğer kullanıcı girişi başarısızsa giriş sayfasına yönlendirir, başarılı ise devam etmesine izin verir.

Kullanıcı bir form gönderdi. Formdaki alanları kontrol eder. Formdan gelen güvenlik ihlali sağlayacak veri varsa onları filtreler ve kalan veriyi işlemesi için bir sonraki aşamaya gönderir.

**ROOT\_URLCONF**  Django ya istek geldiğinde url maping için hang dosyanın kullanılacağını ayarlamaya yarar.

**TEMPLATES** : Template motoru ayar değişikliklerinin yapıldığı bölüm

**DATABASES** : Veritabanı ayar değişikliklerinin yapıldığı bölüm. Django birden fazla veritabanı ile işlem yapmamıza izin vermektedir.

**AUTH\_PASSWORD\_VALIDATORS** : Kullanılacak kullanıcı şifreleri için gerekli olan validasyon ayarları

**LANGUAGE\_CODE**: Django dil ayarı tr yaparak türkçeleştirebiliriz.

**TIME\_ZONE** : Zaman ayarı. Europe/Istanbul yaparak türkiye saatini ayarlayabiliriz.

**USE\_I18N, USE\_L10N, USE\_TZ** : Localizasyon ayarlarıdır.

**STATIC\_URL**: statik dosyaların konunlandıracağı yeri gösterir.

Projemizi oluşturmuş olduk, projemiz uygulamalardan oluşmaktadır. Uygulama oluşturarak devam edelim.

Öncelikle bir tane learning adında uygulama oluşturalım.

**python manage.py startapp learning**

Uygulama oluşturduğumuzda Django otomatik olarak bazı klasör ve dosyaları oluşturmaktadır. Şimdi bu dosyalar ne işe yaradığına kısaca bakalım. İleriki konularda ayrıntılarına gireceğiz. Şimdilik ne işe yaradıklarını bilmemiz yeterli.

**\_\_init\_\_.py :** dosyası bulunduğu klasörün bir PAYTON modülü olduğunu göstermektedir.

**admin.py :** uygulama için django admin panelini özelleştirmeye yarayan dosyadır.

**apps.py :** uygulamanın configürasyon ayarları için kullanılan dosyadır.

**migrations/ :** models.py de yer alan veri yapılarımızın veritabanına taşınmasını sağlayan dosyadır.

**models.py :** veri yapılarımızın tanımlandığı dosyadır. Veritabanında kullanmak istediğimiz veri yapılarını bu dosyalar içerisinde tutarız.

**tests.py :** birim testlerinin yazıldığı dosyadır. Testler kodumuzun çalıştığını kontrol ederler. Bu sayede her değişiklikte manuel test yapmamıza gerek kalmaz.

**views.py :** Tarayıcıdan bir adrese girildiğinde projenin urls.py dosyasından bu dosyanın içerisindeki methodlara istek gönderilir. Bu methodların içerisindeki kodlar çalışarak tarayıcıya bir çıktı response edilir. Daha sonra tarayıcı o çıktıyı ekrana basar.

Dosyalar ile ilgili temel bilgiler bu şekilde olup, uygulama aşamasına geçildiğinde daha iyi anlaşılacaktır.

Bu dosyaları ve klasörleri kullanmak zorunda değilsiniz. Ben büyük projeler için daha kolay okunabilirlik olması nedeniyle view ve modeller için ayrı klasörler kullanıyorum. İki kullanım içinde örnekler yapacağım. Öncelikle klasörler kullanarak oluşturulan yapıyı inceleyelim.

Eğer uygulamalarınız büyükse views.py ve models.py dosyalarının içerisinde çok fazla kod oluşmakta

bu da dosyada yapılan düzenlemeleri ve okunabilirliği zorlaştırmaktadır.

Uygulamada 10 tane model ve her modelinde yaklaşık 30 satır olduğunu düşünürsek models.py dosyası

300 satırdan oluşacaktır. Bu da dosya içerisinde istediğimiz modeli bulmayı ve düzenlemeyi zorlaştıracaktır.

Ben bunun yerine models ve views klasör halinde tutup içerisine her model ve view için dosya oluşturuyorum. Models/ klasörüne baktığımızda product ve product\_detail için iki ayrı dosya oluşturdum ve içerisinde ilgili veri yapılarımı ekledim.

Bu videomuzda uygulama oluşturduk ve uygulama dosyalarını inceledik.

Bir sonraki videomuzda nasıl model oluşturacağımızı öğreneceğiz ve model özelliklerini inceleyeceğiz.

Bu videomuzda Django kurulumunu gerçekleştirdik ve proje ve uygulamanın temellerine değindik.