Arkadaşlar merhaba,

Bir önceki bölümde Modellere giriş yaptık. Nasıl veri yapıları oluşturacağımızı ve veritabanı ile nasıl eş zamanlayacağımızı öğrendik. Oluşturduğumuz veri yapılarının yönetici sayfasında nasıl görüntüleneceğini ve yönetici sayfasının özelleştirilmesi hakkında temek bilgilere değindik. Bu videomuzda temel bilgiler vermeye devam edeceğiz. İleriki konularda ise detaylandıracağız. Bu vidomuzda URL Configuration, Views ve templateler hakkında bilgiler vereceğim.

Kullanıcı tarayıcıyı açtı sitemizin adresini yazdı ve entere bastı. İsteği server Django uygulamasına yönlendiriyor. Django uygulaması istek yapılan adrese ile bizim settings.py den ayarlamış olduğumuz URL Configuration yaptığımız dosyayı karşılaştırır ve hangi URL ile eşleşiyorsa o URL’ye karşılık gelen view metoduna yönlendirir. Biz Django’da URL configurasyonunu stocks/urls.py dosyasında yaparız. Şimdi URL configurasyonu yaptığımız dosyamızı inceleyelim.

Gördüğünüz gibi burada admin arayüzü hazır geliyor. Daha önceki derslerimizde <http://127.0.0.1:8000/admin/> olarak girdiğimizi hatırlarsınız.

Biz buraya example/ isteği yapıldığında başka bir yere yönlendirme yapabiliriz. <http://127.0.0.1:8000/example/> yapıldığında example\_view yönlendirmiş olduk. Tabi yönlendirdikte example\_view nerde ?

urlpatterns = [  
 path('admin/'**,** admin.site.urls)**,** path('example/'**,** views.example\_view)  
]

Gördüğünüz gibi Django’da bulamıyorum ben diye NameError hatası verdi.

**NameError: name 'example\_view' is not defined**

Biz view metodlarımızı views.py dosyasında oluşturuyoruz. Learning uygulamamızda example\_view adında bir tane view metodu oluşturalım. View metodları ilk parametre olarak request değerini alır. Bu parametre ile post, get yapılan verilere ve bir takım diğer verilere ulaşabiliriz.

Html değişkenine ekrana basılacak içeriğimizi girelim. Şimdi bunu ekrana basmamız gerekli bunun için Django http kütüphanesindeki HttpResponse metodunu kullanabiliriz.

from django.http import HttpResponse  
  
  
def example\_view(request):  
 html = "<html><body><h2>MERHABA DÜNYA!</body></head>"  
  
 return HttpResponse(html)

Evet view metodumuz hazır urls.py mizde de tanımlı. Ama hala name error hatası alıyoruz. Çünkü bu metodumuzu urls.py dosyasına import etmedik. İmport edelim ve serverımızı tekrar çalıştıralım.

from learning.views import example\_view

**python manage.py runserver**

Evet serverımız sorunsuz çalıştı. Adresimizi kontrol ettiğimizde ekrana MERHABA DÜNYA! Gelmesi gerekli. Evet oda gelmiş. Yani bir url configürasyonu yaptık ve example/ görürsen example\_view methodunu çalıştır dedik.

Tüm html kodlarını view dosyası içerisine yazmamız çok mantıklı değil. Django’nun MVT yapısına sahip olduğunu söylemiştik. HTML kodları için Template kullanabiliyoruz. View’den templati çağıralım ve ekrana ordan gelen içeriği basalım. Bunun için shortcuts kütüphanesinden render komutunu kullanıyoruz.

from django.shortcuts import render  
  
  
def example\_view(request):  
 return render(request=request**,** template\_name='example/detail.html')

settings.py den template klasörü olarak templates ayarlayalım.

TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates'**,** 'DIRS': ['templates']**,** 'APP\_DIRS': True**,** 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug'**,** 'django.template.context\_processors.request'**,** 'django.contrib.auth.context\_processors.auth'**,** 'django.contrib.messages.context\_processors.messages'**,** ]**,** }**,** }**,**]

Templates ve example klasörlerini oluşturalım ve içerisine detail.html ekleyelim. İsteğimizi yapalım. Evet template dosyasından yazımız geldi.

Peki biz bütün URL Configürasyonumuzu bu ana dosyamızdamı yapacağız. İstersek yapabiliriz fakat her uygulama için farklı bir url dosyası kullanmak daha derli toplu olacaktır. Yapılan istek uygulamanın adıyla yapılıyorsa direk uygulamaya aktaralım. Yani 127.0.0.1:8000/learning/ adresine yapılan istekler learning/urls.py dosyasındaki URL Configürasyonuna yönlendirilsin. Öncelikle uygulamamız içerisinde urls.py dosyamızı oluşturalım ve bu dosyamızı ana konfigürasyon dosyamıza include edelim. Bunun için urls kütüphanesindeki include metodunu kullanacağız. learning/ gelen istekleri learning.urls yönlendirelim.

Evet yine bir hata ile karşılaştık. Learning.urls dosyaımızın doğru şekilde konfigüre edilmediğini söylüyor. Dosyamızı açalım ve example/ gelen isteği example\_view metoduna yönlendirelim ve serverımızı tekrar çalıştıralım.

from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('example/'**,** views.example\_view)  
]

**python manage.py runserver**

serverımız hatasız olarak çalıştı. learning/example/ istek yapıldığında metodumuz çalışıyormu kontrol edelim. Evet gördüğünüz gibi ekrana MERHABA DÜNYA! basıldı.

Model dosyalarımızı klasör içerisinde kullanmıştık. View dosyalarımızıda aynı şekilde kullanmak daha derli toplu olmasını sağlayacaktır. View adında klasör oluşturalım ve views.py dosyasını onun içine alalım. İsminide example olarak değiştirelim. klasörün içine \_\_init\_\_.py dosyaımızı oluşturalım ve example\_view metodunu bırada tanımlayalım.

from .example import example\_view

learning/example adresini kontrol ettiğimizde çalıştığını görüyoruz. Bu sayede view yapımızı hazırlamış olduk.

İstek geldiğince öncelikle ana urls.py dosyamıza gelecek daha sonra burdan ilgili uygulamanın urls.py dosyasına yönlenecek. Daha sonra uygulamanın ilişkili metodu bulunacak ve çalıştırılacak.

Farklı url patternler oluşturalım ve bunlara karşılık view metodlarını yazalım. Learning/views içerisine product.py dosyası oluşturalım ve Product modeli ile ilgili view dosyalarımızı burda yazalım.

# learning/views/product.py

def view\_products(request):  
 pass

metodumuzu \_\_init\_\_.py de tanıtalım.

# learning/views/\_\_init\_\_.py

from .product import view\_products

products/ yapılan istekleri metodumuza yönlendirelim.

# learning/urls.py

path('products/'**,** views.view\_products)**,** # views.view\_products(request)

evet artık products/ istek gönderilince view\_products metodumuza gönderilecek. Şimdi view\_products metodumuzu ürünleri getirecek şekilde ayarlayalım. Product modelimizi import edelim. Ürün listesi vereceği için tüm ürünleri alalım. Sonra içerisindeki name ve author\_name alanlarını kullanalım.

# learning/views/product.py

from django.http import HttpResponse  
from learning.models import Product  
  
def view\_products(request):  
  
 products = Product.objects.all()  
  
 data = []  
 for product in products:  
 item = {  
 'name': product.name**,** 'author\_name': product.author.username  
 }  
 data.append(item)  
  
 return HttpResponse(data)

Evet artık vertabanından veri çekip ekrana basabiliyoruz. select\_related ile birden fazla sorgu yapmasını önleyebiliriz. values() ile istediğimiz alanları alabiliriz.

from django.http import HttpResponse  
from learning.models import Product  
  
def view\_products(request):  
 products = Product.objects.select\_related('author').values('name'**,** 'content'**,** 'author\_\_name')  
 return HttpResponse(list(products))

Şimdi bir tane de ürün detayları için view metodu oluşturalım ve \_\_init\_\_.py’de tanıtalım.

# learning/views/product.py

def view\_product(request):  
 pass

# learning/views/\_\_init\_\_.py

from .product import view\_products, view\_product

product/ yapılan istekleri metodumuza yönlendirelim.Fakat ürünü seçebilmek için bir parametreye ihtiyacımız oalacak primary\_key ile filtreleyelim.

from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('example/'**,** views.example\_view)**,** path('products/'**,** views.view\_products)**,** # views.view\_products(request)  
 path('product/<pk>/'**,** views.view\_product)**,** # views.view\_product(request, pk=1)  
  
]

Böyle bir parametre aldığını view metoduna haber verelim ve console primary\_key bastıralım. Response yapmadığımız için hata verecektir fakat biz consola primary key basıyormu diye kontrol edeceğimiz için önemli değil.

def view\_product(request**,** pk=None):  
 print(pk)

http://**127.0.0.1**:**8000**/learning/product/**5**/  
[**21**/Apr/**2019 02**:**05**:**42**] "GET /product/5/ HTTP/1.1" **404 10858  
5**

Evet ekrana bastığını görebildik. Birde string girerek deneyelim. Stringide bastı. Bizim primary\_key alanımız sadece sayıları kabul ediyor. Buraya sadece sayı nasıl kabul edebiliriz.

http://**127.0.0.1**:**8000**/learning/product/emre/   
[**21**/Apr/**2019 02**:**05**:**52**] "GET /learning/product/5/ HTTP/1.1" **500 68992**emre

Django URL Configürasyonunda <int:pk> yaparak bunu halledebiliriz. Değiştirip tekrar deneyelim. Evet string girdiğimizde Page Not Found hatası verdi. <int:pk> olarak kullanıma path converters adı verilmektedir. Djangoda standart olarak kullanılabilecek path çeviriciler:

* str : sadece string kabul eder ve path seperatörü kabul etmez.
* int : sıfır ve pozitif sayıları kabul eder
* slug : sadece slug kullanımına uygun karakterleri kabul eder
* uuid : uuid formatı kabul eder.
* path : sadece string ve path seperatörü kabul eder.

İstersek kendi path convertörlerimizi register\_converter() metodu ile tanıtabiliriz. Bu konu hakkında detaylı bilgi için djangonun dökümantasyonundan bilgi alabilirsiniz.

path converterlar yerine regex’te kullanılabilir. Bunun için re\_path import etmemiz gerekmektedir.

Sadece sayı alması için;

from django.urls import path**,** re\_path  
from . import views  
  
urlpatterns = [  
 path('example/'**,** views.example\_view)**,** path('products/'**,** views.view\_products)**,** # views.view\_products(request)  
 path('product/<int:pk>/'**,** views.view\_product)**,** # views.view\_product(request, pk=1)  
 re\_path(r'^re\_product/(?P<pk>[0-9])/$'**,** views.view\_product)**,** # views.view\_product(request, pk=1)  
]

Son olarak primary key ile view metodumuza ulaşmıştık ve primary keyin sadece sayı almasını sağlamıştık. Veritabanımızdan o primary key a ait ürün bilgilerini getirelim. Ürün bilgilerimizi template gönderelim ve ordan gelen içeriği ekrana basalım. Bunun için shortcuts kütüphanesindeki render komutunu kullanıyoruz. Template dosyasında ürünlere ulaşabilmek için product’u context argümanı olarak ekleyelim.

from django.shortcuts import render  
  
  
def view\_product(request**,** pk=None):  
 product = Product.objects.select\_related('author').get(pk=pk)  
 return render(request=request**,** template\_name='product/detail.html'**,** context={'product': product})

urlpatterns = [  
 path('example/'**,** views.example\_view)**,** path('product/'**,** views.view\_product)**,** path('product/2010/'**,** views.view\_products\_2010)**,** path('product/<int:year>/'**,**views.products\_year\_archive)**,** path('product/<int:year>/<int:month>/'**,**views.products\_month\_archive)**,** path('product/<int:year>/<int:month>/<slug:slug>'**,** views.product\_detail)  
]