Arkadaşlar merhaba bu videomuzda Class Based Viewsleri öğreneceğiz.

Biz şimdiye kadar viewlerimizi method bazında yani function based views olarak yazdık. Urls.py dosyamızdan her metoda bir url tanımladık. Şimdi bu viewlerimizi class olarak nasıl yazacağımızı öğreneceğiz.

Bunlardan hangilerini kullanmalıyım sorusunun bir cevabı yok. İnternette araştırırsanız bir kısmın Function Based View leri bir kısmın ise Class Based View leri savunduğu göreceksiniz. Function Based View ler ile kod yazmak daha kolay. Class Based Viewler ise daha hızlı kod yazabilirsiniz. özellikle CRUD işlemler için. Tabi Function Based Viewlere göre daha karışık bir yapısı var. Sadece birini kullanmak zorunda da değilsiniz bir kısmı CBW bir kısmı ise FBW ile de yapabilirsiniz.

Genellikle FBW kullanan sadece crud işlemleri ve alt sınıfları olan viewler için CBW kullananlarda var.

Şimdi biz daha önceden Function Based olarak yazdığımız viewleri Class Based viewlere çevirelim. Bu sayede aradaki farkı da görebiliriz. Class Based Viewler django.views kütüphanesinde View sınıfından türerler. Öncelikle View sınıfını import edelim

ProductView adında yeni bir sınıf oluşturalım ve View sınıfından extend edelim. Class Based Viewler request olarak gelen methoda göre sınıfın get, post sınıflarına yönlendirilir. Html ile http methodu olarak post, get ve head gönderilebilir.

from django.views import View

class ProductView(View):  
  
 def get(self**,** request):  
 pass  
   
 def post(self**,** request):  
 pass

from django.views import View

from django.shortcuts import render

class ProductView(View):  
  
 def get(self**,** request):  
 product\_list = Product.objects.order\_by('-name')[:**5**]  
  
 context = {  
 'products': product\_list  
 }  
  
 return render(request=request**,** template\_name='product/list.html'**,** context=context)

def post(self**,** request):  
 pass

En temel class based view sınıfı View’dir. View sınıfından türeyen ListView ve DetailView model ile etkileşimi bir şekilde çalışırlar. Uygulamamıza generic kütüphanesinden ListView ve DetailView sınıflarını import edelim.

Şimdi ürün listeleme sayfamızı ListView ile yapalım.

from django.views.generic import ListView**,** DetailView  
  
  
class ProductListView(ListView):  
 model = Product  
 template\_name = 'product/list.html'

Sadece iki satır kod ile ürünlerimizi listelemiş olduk. ListView’den veriler object\_list olarak döner. Template de products olan yeri object\_list olarak değiştirmeliyiz..

{% for product in object\_list %}

{% endfor %}

Eğer Products olarak kullanmak istersek context\_object\_name ayarlamalıyız.

class ProductListView(ListView):  
 context\_object\_name = 'products'  
 model = Product  
 template\_name = 'product/list.html'

Ürün detay sayfamızıda DetailView ile oluşturabiliriz.

class ProductDetailView(DetailView):  
 model = Product  
 template\_name = 'product/detail.html'

ListView sınıfımızı biraz özelleştirelim. get\_queryset metodu ile sorgu setimizi özelleştirebiliriz. 5 ürün ile sınırlayalım.

class ProductListView(ListView):

context\_object\_name = 'products'  
 model = Product  
 template\_name = 'product/list.html'

def get\_queryset(self):  
 return Product.objects.order\_by('-name')[:**5**]

queryset attribute olarak da yazabiliriz.get\_context\_data ile return edilecek context üzerinde değişiklik yapabiliriz.

class ProductListView(ListView):  
 context\_object\_name = 'products'  
 model = Product  
 template\_name = 'product/list.html'  
 queryset = Product.objects.order\_by('-name')[:**2**]

def get\_context\_data(self**,** \***,** object\_list=None**,** \*\*kwargs):  
 context = super().get\_context\_data(\*\*kwargs)  
 context['new\_field'] = 'new field';  
 return context

Uygulamamızı geliştirirken farklı türde class based viewlerde kullanacağız. <https://ccbv.co.uk/> adresinden kullanılabilecek view listesine, metodlarına vb. detaylı bilgilerine ulaşabilirsiniz.