## در این دوره با نوع دیگری از viewها در جنگو با عنوان Class-based views آشنا میشوید. در حالت عادی viewهایی که در جنگو وجود دارند به شکل یک فانکشن هستند که با کلمه کلیدی def ایجاد میشوند که به آنها Function-based views یا FBV گقته میشود. اما CBVها یک روش دیگر برای ساخت viewها براساس کلاس ها هستند. CBVها قرار نیست که FBVها را از رده خارج کنند بلکه قرار است در شرایط خاص در کنار FBVها باشند. امتیازی که CBVها نسبت به FBVها دارند، کنترل درخواست های HTTP در متدهای مختلف و همچنین استفاده از ارثبری چندگانه برای گسترش استفاده پذیری کد است.

<https://docs.djangoproject.com/en/3.1/ref/class-based-views/>

<https://docs.djangoproject.com/en/3.1/ref/class-based-views/flattened-index/>

================================================================================

در این قسمت با کلاس View در جنگو آشنا میشوید. View از نوع base و ساده ترین کلاسی هستش در جنگو وجود داره. از این کلاس زمانی استفاده کنید که میخواید کنترل کاملی روی هندل کردن درخواست کاربر داشته باشید.

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/base/#view](https://docs.djangoproject.com/en/3.1/ref/class-based-views/base/" \l "view)

تمام وظیفه ی هندل کردن requestها بر عهده ی خودتان میباشد

================================================================================

در این قسمت با کلاس TemplateView در جنگو آشنا میشید. این کلاس یه مرحله بیشتر از کلاس View وارد جزئیات میشه و همونطور که از اسمش مشخص هستش میتونه برای شما یک صفحه html رو render کنه. تنها کاری که نیازه شما داخل این کلاس انجام بدین نوشتن اسم template هست که میخواین کاربر به اون فرستاده بشه. همچنین میتونید با بازنویسی متد get\_context\_data به اون صفحه پارامتر ارسال کنید. از این کلاس زمانی استفاده کنید که میخواید یک صفحه ساده رو به کاربر نشون بدین. اگر قراره داخل صفحه html از فرم ها استفاده کنید یا بخواید یک موردی رو حذف یا آپدیت کنید استفاده از این کلاس پیشنهاد نمیشه.

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/base/#templateview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/base/" \l "templateview)

یک راه خیلی ساده‌ای که میتوان از این class base view استفاده کرد :

from django.views.generic.base import TemplateView   
app\_name = 'first'  
urlpatterns = [  
 path('', TemplateView.as\_view(template\_name='first/home.html'), name='home'),  
]

================================================================================

کلاس ListView در جنگو میتونه اطلاعات رو از داخل یک model خونده و در یک template لیست کنه.

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-display/#listview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-display/" \l "listview)

class Home(ListView): # first/todo\_list.html  
 template\_name = 'first/home.html' # queryset = *Todo.objects.all()*  
 context\_object\_name = 'todos' # object\_list  
 ordering = ['-created']  
  
 def get\_queryset(self):  
 return Todo.objects.all()

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/flattened-index/#listview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/flattened-index/" \l "listview)

به طور پیش‌فرض listview در template زیر مقادیر را ارسال میکند :

name\_app/name\_model\_list.html ====> first/todo\_list.html

================================================================================

از کلاس DetailView میتونید برای خوندن اطلاعات جزئی هر آبجکت از model ها استفاده کنید

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-display/#detailview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-display/" \l "detailview)

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/flattened-index/#detailview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/flattened-index/" \l "detailview)

نکته بسیار مهمی که باید به آن توجه کرد این است که در pathی که به class detail view اشاره میکند باید حتماً یا ازpk یا slug استفاده کنید (detail view انتظار دارد یا pk یا slug به آن ارسال کنید و اگر id ارسال کنید با مشکل مواجه می‌شوید )

template پیش‌فرض هم ماننده list view میباشد با این تفاوت که در آخر بجای list\_ از detail\_ استفاده می‌شود

در detail view هم میتوانید از query\_set استفاده کنید فقط دقت شود که model باید پاک شود

دقت شود که در list و detail اگر قرار است از فرم‌ها استفاده کنید و فرم‌ها را تغییر دهید از این موارد استفاده نکنید چون موارد بهتری وجود دارد

========================================================================

کلاس FormView برای هندل کردن یک فرم بکار میرود. داخل این کلاس شما به یک متد به نام form\_valid نیاز دارید که در صورتی که کاربر از طریق متد post وارد شد و اطلاعاتی که وارد شده معتبر بود این متد فعال خواهد شد. پیشنهاد میشود از کلاس FormView زمانی استفاده شود که فرم نیازی به ارتباط با model نداشته باشد. در صورت نیاز به تغییر model بهتر است از کلاس‌های CreateView , UpdateView و DeleteView استفاده کنید

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-editing/#formview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-editing/" \l "formview)

بجای این کلاس میتوانید از کلاس view هم استفاده کنید اما باید کار های زیادی رو انجام دهید :

class TodoCreateView(View): def get(self): pass def post(self): pass

داخل cbvها اگر بخواهید یک مورد را redirect کنید بهتر از reverse\_lazy استفاده کنید reverse\_lazy حدوداً ماننده revese میباشد دقیقاً معادل تگ url در templateها در خود سایت django هم درباره reverse\_lazy توضیح داده است که این مورد زمانی مناسب است که هنوز urlconf پروژه تون کامل load نشده است و به درد cbvها میخورد [https://docs.djangoproject.com/en/3.1/ref/urlresolvers/#reverse-lazy](https://docs.djangoproject.com/en/3.1/ref/urlresolvers/" \l "reverse-lazy)

در این cbv به یک متد نیاز داریم که زمانی فعال می‌شود که اطلاعاتی که از سمت کاربر آمده باشد از طریق متد post باشد (form\_valid) دقت کنید که حتماً باید form\_valid کلاس والد را هم صدا بزنید

========================================================================

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-editing/#createview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-editing/" \l "createview)

در این کلاس CreateView نسبت به کلاس formview کد کمتری را باید بنویسیم هماننده formview باید form\_valid کلاس والد هم در آخر صدا بزنید

========================================================================

DeleteView این کلاس برای حذف کردن یک آبجکت از model استفاده میشود. برای استفاده از این کلاس باید id آبجکت را از طریق url به این کلاس ارسال کنید. در این حالت کلاس DeleteView یک صفحه html به کاربر نشان داده و از کاربر درخواست تایید عمل حذف کردن را میکند. این صفحه html اجباری هست و template\_nameی که در کلاس وجود دارد اشاره به همین صفحه ی تاییده دارد

objectی که در template استفاده شده است از طریق این cbv قابل دسترس است

[https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-editing/#deleteview](https://docs.djangoproject.com/en/3.1/ref/class-based-views/generic-editing/" \l "deleteview)

========================================================================

UpdateView این کلاس برای آپدیت کردن آبجکت ها بکار میرود. این کلاس به طور خودکار از model یک فرم ساخته و فیلدها را با مقادیر اولیه پر میکند.

========================================================================

LoginView این کلاس برای لاگین کردن کاربرها استفاده میشود. این کلاس به صورت پیشفرض یک فرم را به کاربر نشان داده عمل لاگین کردن را انجام میدهد.

https://docs.djangoproject.com/en/3.1/topics/auth/default/

[https://docs.djangoproject.com/en/3.1/topics/auth/default/#module-django.contrib.auth.views](https://docs.djangoproject.com/en/3.1/topics/auth/default/" \l "module-django.contrib.auth.views)

جنگو به طور پیشفرض در path برای login و logout و .... ازاین الگو استفاده میکند :.../accounts

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

به طور کلی ابتدا باید از این path در url اصلی پروژه استفاده کنید :

urlpatterns = [

path('accounts/', include('django.contrib.auth.urls')),

]

وقتی از این path استفاده میکنیم جنگو تمام موارد رو هندل میکند به خاطر همین دسترسی بسیار کمی برای شخصی سازی داریم برای همین بهتر است در این موارد خود موارد رو هندل کنیم

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

در این موارد جنگو به طور پیش‌فرض مسیرtemplate هارا خود تعیین میکند اما میتوانیم مسیر template را خود دستی تعیین کنیم به صورت زیر :

app\_name = 'accounts'  
urlpatterns = [  
 path('login/', views.UserLogin.as\_view(template\_name=’accounts/login.html’), name='login'),  
 .........  
]

یا میتوانیم ماننده دیگر cbv ها رفتار کنیم که یک کلاس در داخل views.py ساخته شود

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

بعد از اینکه login با موفقیت انجام شد به طور پیشفرض به صفحه accounts/profile هدایت می‌شود اگر بخواهید این پیشفرض را تغییر بدهید میتوانید در داخل settings.py مقدار LOGIN\_REDIRECT\_URL را قرار دهید

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

مورد خوبی که وجود دارد extra\_context میباشد و به صورت زیر استفاده می‌شود

extra\_context = {‘name’ : ’sajjad’}

حال در template میتوانید از name استفاده کنید

=============================================================================