## 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)

from django.views import View

class MyView(View):

def get(self, request, \*args, \*\*kwargs):

return render(request, 'first.html', {‘name’:”sajjad”})

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

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

CreateView

[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

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

LogoutView   
این کلاس برای عمل logout کاربر استفاده میشود.

به طور پیش‌فرض جنگو برای logout کردن کاربر را به LOGOUT\_REDIRECT\_URL هدایت میکند که دوباره به طور پیشفرض None میباشد شما میتوانید از روش دیگری هم برای redirect کردن نیز استفاده کنید که هر دو روش در زیر مشاهده میکنید :

1) in settings.py ====> LOGOUT\_REDIRECT\_URL = ‘first:home’

2) path('logout/', auth\_views.LogoutView.as\_view(next\_page = ‘first:home’), name='logout'),

همین‌طور میتوانید ماننده بقیه CBVها یک template\_name به این cbv بدهید

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

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

برای فرستادن ایمیل فراموشی رمز عبور به کاربران باید از یک سرویس دهنده ایمیل استفاده کنیم. ما برای این کار از جیمیل استفاده خواهیم کرد.

https://docs.djangoproject.com/en/3.1/ref/settings/

برای استفاده از ایمیل در پروژه لازم است یک سری تغییرات در فایل settings.py انجام دهید :

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

email\_backend به طور پیش‌فرض هم همین مقدار را دارد

EMAIL\_HOST = ‘smtp.gmail.com’

EMAIL\_HOST\_USER = ‘[mr.ahmadalimoshtaghi@gmail.com](mailto:mr.ahmadalimoshtaghi@gmail.com)’

EMAIL\_HOST\_PASSWORD = ‘jdkfhdskfhdskhfdsjk’

این همان مقداری است که بعد از فعال سازی gmailتان برای ارسال ایمیل دریافت میکنید

EMAIL\_PORT = 587

به طور کلی اگر از جیمیل استفاده میکنید باید ۵۸۷ را انتخاب کنید

EMAIL\_USE\_TLS = True

به طور پیش‌فرض false است اما یک راه ارتباطی امن به gmail برقرار میکند پس بهتر است آنرا true قرار دهید (EMAIL\_USE\_SSL هم وجود دارد که پیشنهاد جنگو استفاده از TSL میباشد)

DEFAULT\_FROM\_EMAIL = ‘Ahmad Ali <ahmad.ali@gamil.com>’

این مورد را به بالای ایمیل شما میچسباند

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

PasswordResetView   
زمانی کاربر رمز عبور خود را فراموش کرده باشد این کلاس یک input به کاربر نشان داده از کاربر باید ایمیل خود را وارد کند. در صورت درست بودن ایمیل کاربر، یک ایمیل حاوی لینک تغییر رمز عبور به کاربر ارسال می‌شود

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

تقریباً در این cbv باید ۳ مقدار را مقدار دهی کنید :  
template\_name که htmlی هست که کاربر باید به آن برود تا ایمیلش را وارد کند   
success\_url که htmldی هست که بعد از ارسال موفقیت آمیز ایمیل کاربر را به این صفحه هدایت میکند   
email\_template\_name که فایل htmlی میباشد که میخواهید به کاربر ارسال کنید و آن را ببیند \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

در html\_email\_template\_name حتماً باید کد زیر را قرار دهید چرا که همان لینک ریست پسورد میباشد

{{ protocol}}://{{ domain }}{% url 'password\_reset\_confirm' uidb64=uid token=token %}

دقت کنید که در تگ url باید آدرس path کی مربوط به confirm میباشد را قرار دهید در این پروژه به این شکل میباشد :{{ protocol}}://{{ domain }}{% url 'accounts:password\_reset\_confirm' uidb64=uid token=token %}  
  
در template که برای اینcbv ایجاد کردیم میبتوانیم با استفاده از {{email}} به ایمیلی که در خواست ریست پسورد کرده است را ببینیم   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PasswordResetDoneView  
بعد از اینکه در مرحله قبل یک لینک به کاربر ایمیل شد، کلاس PasswordResetDoneView یک صفحه html به کاربر نمایش خواهد داد و در آن پیغام موفقیت آمیز بود ارسال ایمیل به کاریر نشان داده میشود

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

این همان success\_urlی هست که در cbv بالا نوشتیم و به این کلاس اشاره میکند

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

PasswordResetConfirmView  
 در CBVهای قبل تونستیم یک لینک به کاربر ایمیل کنیم. بعد از اینکه کاربر روی لینک کلیک کرد کلاس PasswordResetConfirmView یک فرم به کاربر نشان داده و کاربر میتواند رمز عبور جدید خود را وارد کند.

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

در این CBV باید دو مورد را از طریق path ارسال کنید :  
uidb64 , token میباشد این مورد چیزهایی است که جنگو ست میکند تا کاربر را شناسایی کند

حال در خوده این CBV باید چندین مقدار مقداردهی شوند :  
template\_name فایل htmlی که قرار است فرم مورد نظر را در آن ببیند

در این cbv دو مورد context وجود دارد .اولین مورد همون form میباشد که در تمام cbv هایی که تاکنون خوانده‌ایم هم وجود داشته‌اند. دومین مقدار validlink میباشد که boolean میباشد که چک میکند که این لینکی که کاربر باهاش وارد شده معتبر هست یا نه (قبلاً استفاده نشده باشد)

توجه شود که باید success\_url مقدار دهی شود و به path کلاس PasswordResetCompleteView اشاره کنند

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

PasswordResetCompleteView  
بعد از اینکه کاربر در مرحله قبل رمز عبور جدید خود را وارد کرد، به کلاس PasswordResetCompleteView ارجاع داده خواهد شد. این کلاس یک پیغام مبنی بر کامل شدن فرآیند تغییر رمز عبور به کاربر نشان خواهد داد

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

MonthArchiveView  
این کلاس جزو دسته generic date views میباشد. با استفاده از این کلاس میتوانید آبجکت هایی که در ماه خاصی ایجاد شدند را از دیتابیس بخوانید

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

از این cbv میتوان به دو صورت استفاده کرد به این صورت که این دو حالت در path باهم متفاوت هستند :

path('<int:year>/<int:month>/',  
 ArticleMonthArchiveView.as\_view(month\_format='%m'),  
 name="archive\_month\_numeric"  
 )# Example: /2012/08/

path('<int:year>/<str:month>/',  
 ArticleMonthArchiveView.as\_view(),  
 name="archive\_month"  
 )# Example: /2012/aug/

همانطور که میبینید این تفاوت در مقداری هست که برای ماه در نظر میگیریم (int و srt)

در داخل viewها باید مقادیر زیر را مقدار دهی کنید :

model که مدلی که قرار است از آن اطلاعات خوانده شود   
date\_field که در این مدل چه فیلدی براساس تاریخ است   
month\_format که برابر m% قرار میدهیم (اگر به صورت int باشد اما اگر به صورت str باشد دیگر نیازی به این مورد نمی باشد )  
template\_name اطلاعاتی که خوندی در کدام template نمایش بده(به صورت پیش‌فرض هم آدرسی وجود دارد)

اطلاعاتی که خواند در template ها در object\_list ذخیره می‌شود

در templateها به مواردی همچون previous\_month و next\_month دسترسی داریم

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

mixin   
LoginRequiredMixin  
با استفاده از mixinها میتونید کارهای بیشتری را با کد کمتر انجام بدهید. اولین mixin که با آن آشنا خواهید شد LoginRequiredMixin است که برای محدود کردن دسترسی کاربران لاگین کرده به محتویات است  
یکسری کلاس‌هایی هستند که میتوانید از آن‌ها استفاده کنید که یکسری functionalityها به برنامه تون اظافه کنید

[https://docs.djangoproject.com/en/3.1/topics/auth/default/#the-loginrequired-mixin](https://docs.djangoproject.com/en/3.1/topics/auth/default/" \l "the-loginrequired-mixin)

نکته‌ای که باید به آن توجه کرد login\_url است با توجه به اینکه طبق این پروژه url مربوط به login برابر   
/accounts/login میباشد و به صورت پیش فرض جنگو همین url برای login در نظر گرفته میشود دیگر لازم نیست login\_url مقداردهی شود در غیر اینصورت باید login\_url مقدار دهی شود   
این مقدار را نیز متوانید داخل settings.py مقدار دهی کنید

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

FromMixin  
این کلاس برای زمانی هستش که شما نیاز دارید در کنار یک generic view مثل DetailView از فرم ها هم استفاده کنید

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

در formmixin هنگامیکه که میخواهید import کنید لازم است حتماً edit وجود داشته باشد

مواردی که باید در view هندل شود :  
form\_class فرمی است که باید از آن استفاده شود   
get\_success\_url بعد از اینکه فرم به طور موفقیت آمیز ذخیره شد به کدام url هدایت شود

چون قرار است این فرم از طریق متد post ذخیره شود باید def post ساخته شود برای دسترسی داشتن به فرم در این function میتوانیم از طریق ()self.get\_form به آن دسترسی داشت   
برای دسترسی داشتن به اون objectی که میخواهید روی آن عملیات انجام دهید میتوانید از طریق   
()self.get\_object به آن دسترسی پیدا کرد

دقت شود که حتماً باید form\_valid کلاس والد return شود (return super().form\_valid(form))