BASE\_DIR اشاره به پوشه اصلی پروژه دارد . دقیقا محل ذخیره manage.py

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

برای اضافه کردن محل جدید برای template ها کافیست در فایل settings.py کد زیر به قسمت template اضافه شود

'DIRS': [os.path.join(BASE\_DIR, 'templates')],

به این صورت که پوشه templates در کنار manage.py قرار میگیرد

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

فیلد slug نباید در دیتابیس تکراری باشد چرا که میخواهیم با استفاده از slug آدرس دهی انجام دهیم برای این منظور جنگو optionی دارد به نام unique که اگر برابر True قرار گیرد دیگر اجازه ندارد مورد تکراری ذخیره کند

slug = models.SlugField(max\_length=120, unique=True)

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

تغییراتی که در فایل models.py صورت میگیرد به صورت مستقیم در admin panel اعمال میشود و لازم نیست migrate انجام شود

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

from django.utils import timezone

این برای جنگو میباشد و برای مقدار دهی به این صورت اعمال میشود

publish = models.DateTimeField(default=timezone.now)

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

import datetiem

این برای پایتون میباشد و به صورت زیر اعمال میشود

publish = models.DateTimeField(default=datetime.datetime.now)

دقت شود که نباید در آخر now پرانتز گذاشته شود

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

فیلد DateTimeField آپشن هایی دارد به نام های auto\_now\_add و auto\_now

Auto\_now\_add اگر مقدارش True باشد در admin panel دیگر نمایش داده نمیشود و وقتی برای اولین بار دکمه save زده میشود خودکار مقدار دهی میشود یعنی درزمان ساخت مقداردهی میشود و این مقدار غیرقابل تغییر میباشد

Auto\_now اگر مقدارش true باشد همانند auto\_now\_add در admin panel نمایش داده نمیشود اما با این تفاوت که با هر بار زدن دکمه save مقدار عوض میشود یعنی update میشود پس این مقدار با update کردن data مقدارش عوض میشود

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

STATUS = (  
 ('draft', 'Draft'),  
 ('publish', 'Publish')  
)

اولین مقدارها یعنی draft و publish در دیتابیس ذخیره میشودند اما دومین مقدار ها یعنی Draft وPublish نمایش داده میشود

status = models.CharField(max\_length=20, choices=STATUS, default='draft')

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

در بحث on\_delete چندین مقدار وجود دارد که به توضیح هر کدام میپردازیم :

CASCADE: user که حدف شد مقالاتش هم حذف بشه

PROTECT:از user محافظت میکند و اجازه حذف داده نمیشود   
SET\_NULL: اگر یوزر حذف شد مقدار مقالات writer برابربا null بشود

SET\_DEFAULT: یک مقدار پیش فرض داده میشود

SET: مقداری بهش میدهیم

Do\_nothing: هیچ کاری نکن

بهترین آنها دو مورد اول میباشد

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

برای اینکه مدیریت کردن templateها در پروژه های بزرگ ساده تر انجام بشه، داخل هر app یک دایرکتوری به نام templates ایجاد میکنیم و داخل این دایرکتوری یک دایرکتوری دیگه دقیقا همنام اسم app ایجاد میکنیم و فایل های html رو داخل این دایرکتوری قرار میدهیم

در فایل settings.py در قسمت templateها APP\_DIRS = True وجود دارد که به معنی این میباشد که در دایرکتوری خود appها هم به دنبال پوشه templates بگرد

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

زمانی که مدل ها ساخته میشود به صورت پیشفرض جنگو managerی به شما میدهد که objects میباشد

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

Render فانکشنی میباشد که اطلاعت views به template ها فرستاده میشود اولین مقدارش request و دومین مقدارش آدرس template میباشد و سومین مقدار اطلاعاتی است که قرار است به template فرستاده شود

return render(request, 'blog/all\_articles.html', {'all\_articles':all\_articles})

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

در این قسمت با متد get\_absolute\_url در جنگو آشنا میشید. این متد زمانی استفاده میشه که تعداد پارامترهایی که قراره به یک url ارسال بشه زیاد هستش و اگه از تگ {% url %} داخل templateهای جنگو استفاده کنید ممکنه کدمون پیچیده بشه. به همین خاطر در modelها جنگو یک متد به اسم get\_absolute\_url ایجاد میکنیم و پارامترها رو داخل این متد صدا میزنیم. و بعد داخل templateها از آبجکت ها این متد رو صدا میزنیم.

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

Revers دقیقا معادل {% url %} در template ها میباشد (تگ url در فایل main.txt پروژه Django-masterclass توضیح داده شده است )

def get\_absolute\_url(self):  
 return reverse('article\_detail',

args=[self.id, self.slug])

<a href="{{ article.get\_absolute\_url }}">

{{ article.title }}</a>

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

در این قسمت با متد get\_object\_or\_404 در جنگو آشنا میشوید. همینطور که از اسم این متد مشخص هستش این متد سعی میکنه که یک آبجکت رو از دیتابیس بخونه و اگه وجود نداشت صفحه ۴۰۴ نشون میده. در این قسمت همچنین میبینید که چطور میتونید صفحه ۴۰۴ مخصوص به خودتون رو به جنگو بشناسونید

get\_object\_or\_404 اولین مقداری که میگیرد کلاسی هست که میخواهید object مورد نظر را بیابید و دومین مقدار id وslug مورد نظر میباشد

article = get\_object\_or\_404(Article, id=id, slug=slug)

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

برای اینکه بتوانیم صفحه 404 مخصوص به خود را نمایش دهیم باید در فایل settings.py عبارت DEBUG را برابر False قرار دهید . هنگامیکه این کار را انجام میدهید باید در قسمت ALLOWED\_HOST آدرس دامنه ی سرور خود را قرار دهید (بدون پورت )

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

برای نمایش صفحه 404 مخصوص به خود فقط باید در پوشه templateی که در کنار manage.py وجود دارد فایل 404.html ساخته شود

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

اگر در فایل های پایتونی در pycharm تغییر ایجاد کنید خودکار سرور restart میشود اما برای فایل های htmlی باید بصورت دستی سرور را restart کنید تا تغییرات اعمال شود

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

در این قسمت با managerها در جنگو آشنا میشید. managerها یک رابطی هستند که به شما اجازه میدن روی modelهاتون کارهای خاصی رو انجام بدین مثل خواندن اطلاعات، نوشتن اطلاعات، تغییر اطلاعات یا حذف اطلاعات. به صورت پیشفرض هر زمانی که یک model جدید ایجاد میکنید جنگو یک manager به اسم objects در اختیار شما قرار میده که میتونید خیلی از کارها رو با این manager انجام بدین. اما ممکنه شرایطی پیش بیاد که شما بخواید از یک فیلتر پیچیده در جاهای مختلف استفاده کنید. در این حالت میتونید manager خاص خودتون رو ایجاد کنید تا بتونید با این manager کار خودتون رو راحتتر کنید.

برای ساخت manager مخصوص به خود در فایل models.py کلاسی ساخته میشود که باید از models.Manger ارث بری کند

class PublishedArticlesManager(models.Manager):  
 def get\_queryset(self):  
 return super().get\_queryset().filter(

status='publish'

)

در داخل همین کلاس باید متد get\_queryset ساخته شود و هنگام return از get\_queryset منیجر اصلی استفاده شود (super().get\_queryset())

حال باید در مدل خود این manager شناسانده شود

class Article(models.Model):

.....

objects = models.Manager()

published = PublishedArticlesManager()

اکنون دیگر جنگو objects را به عنوان manger اصلی نمیشناسد و باید به مدل شناسانده شود (objects = models.Managet())

برای استفاده از این manager در views هم باید به این شیوه عمل شود

all\_articles = Article.published.all()

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

زمانی که در جنگو urlی ایجاد میکنیم امکان اختصاص دادن یک اسم به url وجود دارد. مشکلی که وجود دارد، اگر تعداد url ها در پروژه زیاد شود، ممکن است اسم urlها تکراری شوند. جنگو برای جلوگیری از این اتفاق استفاده از namespace ها را پیشنهاد میکند.

path('accounts/', include(

'accounts.urls', namespace='accounts')

),

app\_name = 'accounts'  
urlpatterns = [  
 path('login/', views.user\_login, name='user\_login')  
]

href="{% url 'accounts:user\_login' %}

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

گاهی اوقات برنامه خود را تغییر داده اید و میخواهید چک کنید که برنامه ی شما مشکلی دارد یا نه میتوانید از دستور python manage.py check استفاده کنید

این دستور بیشتر به درد syntax error میخورد

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

آبجکت request وضعیت فعالیت کاربر در برنامه ما را مشخص میکند. به طور مثال در آبجکت request با استفاده از اتریبیوت authenticated میتوانید وضعیت لاگین کاربر را بررسی کنید. و یا با استفاده از اتریبیوت session میتوانید به سشن هایی که در مرورگر کاربر ذخیره کرده اید دسترسی پیدا کنید. آبجکت request جنگو در view و template در دسترس است اما در model ها در دسترس نیست.

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

در این قسمت با فرم ها در جنگو آشنا میشوید. برای کار کردن با فرم ها در جنگو میتونید خیلی ساده به شکل سنتی از تگ های html استفاده کنید اما استفاده از تگ های html برای زمانی که بخوایید کارهایی پیچیده تر انجام بدین مشکل خواهد بود. مثلا اگر بخوایید اطلاعات ورودی کاربر رو اعتبار سنجی کنید و یا یک پیغام خاصی رو به کاربر نشون بدین کار برای شما سخت خواهد شد. در عوض جنگو یک کلاس مازول کامل به نام forms در اختیار شما قرار داده که میتونید از اون استفاده کنید. با استفاده از ماژول forms در جنگو میتونید نوع داده ای که هر نوع فیلد قراره از کاربر بگیره رو مشخص کنید و خود جنگو به صورت اتوماتیک براتون اون فرم رو render میکنه.  همچنین اگر نیاز به اعتبارسنجی سفارشی هم داشته باشید خیلی ساده میتونید اینکار رو انجام بدین

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

برای نشان دادن formها ما 3 روش داریم :

Forms.as\_table و forms.as\_p و forms.as\_ul

Forms.as\_ul به عنوان تگ li عمل میکند

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

زمانی که در جنگو با فرم ها کار میکنید، داخل view شما باید با استفاده از یک شرط مشخص کنید که آیا درخواست از طریق متد POST ارسال شده یا متد GET. اگر متد GET باشه به این معنی هستش که کاربر تازه وارد صفحه شده و باید یک فرم خالی نشان داده شود. اما اگر متد POST باشه یعنی کاربر اطلاعات رو در فرم وارد و ارسال کرده، پس ما باید اطلاعات ورودی کاربر رو هندل کنیم.