1.在数据库中使用UTC时区保存时间，是好的做法。避免每年因为Daylight Saving Time所引起的时间推迟提早1小时引起的错误。

USE\_TZ = True将为Django启动时区的支持。并且在存入数据库中的时候，都会带有时区。如果在使用了USE\_TZ但是试图存入naïve的datetime时候，Django会自动根据另一个关键字TIME\_ZONE的设置进行转化。

2.

在django.utils.timezone模块中，

is\_aware(value)方法：判断传入的datetime是否是设置时区的，True为设置时区，False为没有设置时区

is\_naive(value)方法：判断传入的datetime是否没有设置时区，True为没有设置时区，False为设置时区

now () 方法：如果USE\_TZ设置为False则返回一个naïve的datetime系统现在的时间。如果USE\_TZ设置为True, 则返回一个aware的datetime时间，该datetime设置为UTC时区（值根据当地时区转换过来）。

3.

Django项目中有default time zone以及current time zone设置。

Default time zone是通过TIME\_ZONE在settings.py中设置。

current time zone是通过django.utils.timezone.active方法来设置，如果不设置则值默认域default timezone.

4.

当USE\_TZ = True的时候，Django将使用current time zone解析form中填入的时间，并且在form.cleaned\_data中是该时区的aware datetime.

当USE\_TZ = True的时候，Django将使用current time zone来在模板中显示时间。

一个Datetime在Django中的变化流程：

我们在Django中设置USE\_TZ = True, TIME\_ZONE = “America/Los\_Angeles”

-🡪提交表单，从表单中将一个datetime放入数据库中。

用户输入2015/6/10， 8AM然后提交，Django发现我们启动了USE\_TZ = True,并且Time\_ZONE = “America/Los\_Angeles”,另外我们没有额外设置Current time zone.所以Current time zone默认也为Time\_Zone的值。当我们调用form.is\_valid的时候， Django将这个时间字符串变为aware的，并且时区为 “America/Los\_Angeles”. 然后将这个aware datetime放入数据库中（在放入数据库中的格式根据数据库有所不同）。

-🡪 从数据库中取出该时间并且显示在Django模板上

Django根据current time zone将数据库中的aware datetime变为current time zone的格式显示。

5.

时间戳就是Unix时间戳(Unix timestamp)，定义为从格林威治时间1970年01月01日00时00分00秒起至现在的总秒数。然后时间戳是不会跟着时区的改变而改变，所以不管你在哪个时区，时间戳都是一致的。

当谈论时间，几点的时候，必须引入时区。只有在时区存在的情况下才会由几点几分的概念。同样的，从时间戳变为时间或者时间字符串的时候，必须引入时区概念。

例如在Python当中：

statusTimelocal=1599952208 # 时间戳为1599952208

tmp\_time = datetime.fromtimestamp(float(statusTimelocal)) # datetime.datetime(2020, 9, 12, 19, 10, 8)结果显示为19时10分。因为涉及到时，分所以肯定引入了时区的概念。虽然在此没有显式指定时区，但是函数默认使用本地时区作为时区，产生出结果。

Aware的datetime是设置了时区，所以可以方便的转化为UTC。而Naïve的datetime没有设置时区。

6.***Datetime中tzinfo关键字参数设置时区和pytz的不兼容性***

原文：Unfortunately using the tzinfo argument of the standard datetime constructors ‘’does not work’’ with pytz for many timezones. 使用datetime构造器的关键字参数tzinfo和使用pytz够构建的在许多时区会不统一,如果该时区包含DST则无法统一。

例如：对于时间字符串2019-09-15 13:54:24

使用datetime的关键字来构建的时候：

test = “2019-09-15 13:54:24”

test\_datetime = datetime.strptime(test, "%Y-%m-%d %H:%M:%S")

test\_datetime = test\_datetime.replace(tzinfo=pytz.timezone("Canada/Eastern"))

print(test\_datetime) # 为2019-09-15 13:54:24-05:18， timezone为<DstTzInfo 'Canada/Eastern' LMT-1 day, 18:42:00 STD>

使用pytz来构建的时候：

test = “2019-09-15 13:54:24”

test\_datetime = datetime.strptime(test, "%Y-%m-%d %H:%M:%S")

eastern = timezone('Canada/Eastern’)

print(eastern.localize(test\_datetime)) # 为2019-09-15 13:54:24-04:00, timezone为tzinfo=<DstTzInfo 'Canada/Eastern' EDT-1 day, 20:00:00 DST>

基于此，Django在USE\_TZ=True的情况下所进行时间区转化是根据pytz模块的，所以如果自己手动的使用datetime中关键字tzinfo来进行改变时区进行构建，很有可能于Django（基于pytz）对同一时区的时间转化有所不同。