Chrono - Development #71918

Les soucis de date autour des changements heure d'été/hiver viennent de pytz/django

01 décembre 2022 12:59 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	01 décembre 2022
Priorité:	Normal	**Echéance:**	
Assigné à:	Benjamin Dauvergne	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	Non

Description

En voulant m'essayer à implémenter les dates de début de réservation variables, j'ai essayé d'utiliser dateutil.tz au lieu de pytz et je me suis aperçu qu'on avait rien à faire quand on fait des date + timedelta(days=x) pour que l'heure reste la même, ça marche tout seul (pareil avec tzlocal) en cherchant un peu j'ai vu que pytz était totalement déprécié et plus utilisé en Django 4 (en fait django est tout buggé parce qu'il utilise pytz et python 3.6 a introduit un changement non supporté par pytz si j'ai bien compris).

Un code qui essaie de faire un peu pareil que ce qui est fait dans Agenda.max_booking_datetime.

```
$ cat test_date.py
import datetime

import dateutil.tz

from django.conf import settings
from django.utils.timezone import localtime, now

import django

print(django.VERSION)

class Settings:
    TIME_ZONE = 'Europe/Paris'
    USE_TZ = True

settings.configure(Settings)

base_dj = localtime(now()).replace(hour=0, minute=0, second=0, microsecond=0)

europe_france_dtutil = dateutil.tz.gettz('Europe/Paris')

base_dateutil = localtime(now(), europe_france_dtutil).replace(hour=0, minute=0, second=0, microse
cond=0)

for i in range(366):
    dt_dj = localtime(base_dj + datetime.timedelta(days=i))
    dt_dateutil = localtime(base_dateutil + datetime.timedelta(days=i))
    # check dst offset is equal
    assert dt_dj.dst() == dt_dateutil.dst(), (dt_dj.dst(), dt_dateutil.dst())
    if not dt_dj.hour == base_dj.hour:
        print('iteration', i)
        print('django')
        print(base_dj, ' ---> ', dt_dj)
        print('dateutil')
        print(base_dateutil, ' ---> ', dt_dateutil)
        break
 python3 test_date.py
(3, 2, 15, 'final', 0)
iteration 116
django
2022-12-01 00:00:00+01:00  --->  2023-03-27 01:00:00+02:00
dateutil
2022-12-01 00:00:00+01:00  --->  2023-03-27 00:00:00+02:00
```

C'est le deuxième appel à localtime() qui foire la date coté django/pytz, avant ça l'offset dst est mauvais (il n'a pas été ajusté par le + timedelta() après il est bon mais l'heure est faussée). dateutil.tz n'a pas ce comportement, zoneinfo qui est utilisé par django 4 non plus.

Demandes liées:

Lié à Chrono - Development #56284: Idée : avoir un délai de réservation qui s...	**Fermé**	**20 août 2021**

Historique

#1 - 01 décembre 2022 18:27 - Emmanuel Cazenave

C'est bon à savoir.

Pour le plan d'action, je dirais de ne rien faire maintenant, qu'il faudra juste se souvenir de virer le code superflu quand on bossera sur la compat django 4.2 (LTS).

#2 - 01 décembre 2022 18:55 - Benjamin Dauvergne

Le comportement est vraiment parfait, jamais une date qui n'existe pas et autant que possible la bonne heure :
(2ème colonne django, 3ème dateutil)

```
113 2023-03-24 02:00:00+01:00 2023-03-24 02:00:00+01:00
114 2023-03-25 02:00:00+01:00 2023-03-25 02:00:00+01:00
115 2023-03-26 03:00:00+02:00 2023-03-26 03:00:00+02:00
116 2023-03-27 03:00:00+02:00 2023-03-27 02:00:00+02:00
117 2023-03-28 03:00:00+02:00 2023-03-28 02:00:00+02:00

330 2023-10-27 03:00:00+02:00 2023-10-27 02:00:00+02:00
331 2023-10-28 03:00:00+02:00 2023-10-28 02:00:00+02:00
332 2023-10-29 02:00:00+01:00 2023-10-29 02:00:00+02:00
333 2023-10-30 02:00:00+01:00 2023-10-30 02:00:00+01:00
334 2023-10-31 02:00:00+01:00 2023-10-31 02:00:00+01:00

330 2023-10-27 04:01:00+02:00 2023-10-27 03:01:00+02:00
331 2023-10-28 04:01:00+02:00 2023-10-28 03:01:00+02:00
332 2023-10-29 03:01:00+01:00 2023-10-29 03:01:00+01:00
333 2023-10-30 03:01:00+01:00 2023-10-30 03:01:00+01:00
334 2023-10-31 03:01:00+01:00 2023-10-31 03:01:00+01:00
```

#3 - 08 mars 2023 08:28 - Benjamin Dauvergne

- Lié à Development #56284: Idée : avoir un délai de réservation qui soit en journée horaire et pas calendaire ajouté

#4 - 08 mars 2023 08:31 - Benjamin Dauvergne

Emmanuel Cazenave a écrit :

> Pour le plan d'action, je dirais de ne rien faire maintenant, qu'il faudra juste se souvenir de virer le code superflu quand on bossera sur la compat django 4.2 (LTS).

Les bugs sont déjà là en 3.2 et avant, ils ont toujours été là en fait, ça tombe en marche parce qu'on se cale (j'ai fait ça) à midi pour les calculs de date puis qu'on recale à minuit du jour d'avant; sans ça ce serait super visible.

#5 - 08 mars 2023 08:38 - Benjamin Dauvergne

- Assigné à mis à Benjamin Dauvergne

#6 - 08 mars 2023 15:03 - Robot Gitea

- Statut changé de Nouveau à En cours

Benjamin Dauvergne (bdauvergne) a ouvert une pull request sur Gitea concernant cette demande :

- URL : https://git.entrouvert.org/entrouvert/chrono/pulls/46
- Titre : WIP: misc: use zoneinfo instead of pytz for timezones (#71918)
- Modifications : https://git.entrouvert.org/entrouvert/chrono/pulls/46/files

#7 - 08 mars 2023 15:14 - Benjamin Dauvergne

Faut lire https://peps.python.org/pep-0495/ pour comprendre pourquoi c'est bien, la chose principale à savoir c'est qu'il n'y a plus de souci de parsing de date ambiguë ou qui n'existent pas (les gros problème venant de la méthode .localize(dt, is_dst) des timezones pytz).

Le petit changement c'est que les dates pendant un changement d'heure ne sont pas comparable en égalité quand elles ont une timezone.

Le gros avantage c'est que l'arithmétique entre dates devient stable, (x + timedelta(days=n)).hour == x.hour quelque soit n. Dans le cas d'heures qui n'existent pas si on veut une vrai date à afficher il faut faire un roundtrip via UTC x.astimezone(utc).astimezone(localtz).

#8 - 09 mars 2023 00:16 - Robot Gitea

- Statut changé de En cours à Solution proposée

#9 - 14 mars 2023 12:43 - Emmanuel Cazenave

Des précisions (pour que tout le monde puisse suivre, et pour que tu me corriges si je trompe).

Tu te reposes sur le fait que django 3.2 permet déjà l'usage de zoneinfo au lieu de pytz (via https://github.com/django/django/pull/13877).

Django permet ça en acceptant des objets zoneinfo.ZoneInfo, dans les méthodes de django.utils.timezone. Pour qu'on ait pas à s'embêter à ajouter cet argument dans tous nos appels, tu as fait ce petit wrapper chrono/utils/timezone.py.

(tout ça me parait bien)

#10 - 14 mars 2023 13:45 - Benjamin Dauvergne

Emmanuel Cazenave a écrit :

> Des précisions (pour que tout le monde puisse suivre, et pour que tu me corriges si je trompe).
>
> Tu te reposes sur le fait que django 3.2 permet déjà l'usage de zoneinfo au lieu de pytz (via https://github.com/django/django/pull/13877).
>
> Django permet ça en acceptant des objets zoneinfo.ZoneInfo, dans les méthodes de django.utils.timezone. Pour qu'on ait pas à s'embêter à ajouter cet argument dans tous nos appels, tu as fait ce petit wrapper chrono/utils/timezone.py.

Oui. En fait django<4 est fortement dépendant de pytz dans la mesure ou il appelle une API privée de pytz (timezone.localize[1]) qui est celle qui fout le boxon et qui est buggé et qui nous vaut toutes les traces du style AmbiguousDatetime ou NonExistentDatetime qu'on a eu sur wcs ou ailleurs et qu'on a résolu en passant is_dst=True/False qui n'est pas une bonne solution. Avec zoneinfo ou tzlocal, enfin toutes les libs qui ne sont pas pytz et qui respectent la pep-0495 on a plus ces soucis.

Le wrapper m'a semblé la solution la plus simple pour s'assurer que le comportement était uniforme partout et on pourra facilement remigrer vers les APIs standard en django 4. Si on veut corriger les mêmes soucis ailleurs on pourrait copier le module et faire la même opération qu'ici (et virer tous les is_dst=True/False du code[2]).

1

```
# django.utils.timezone 3.2.16

def make_aware(value, timezone=None, is_dst=None):
    """Make a naive datetime.datetime in a given time zone aware."""
    if timezone is None:
        timezone = get_current_timezone()
    if _is_pytz_zone(timezone):
        # This method is available for pytz time zones.
        return timezone.localize(value, is_dst=is_dst)
```

2

```
wcs/qommon/tokens.py:            self.expiration = make_aware(datetime.datetime.fromtimestamp(self.expiration)
, is_dst=True)
wcs/sql.py:                        trace.timestamp = make_aware(datetime.datetime(*evo.time[:6]), is_dst=True)
wcs/sql.py:                            trace.timestamp = make_aware(action[0], is_dst=True)
wcs/workflows.py:            anchor_date = make_aware(anchor_date, is_dst=True)
```

#20 - 14 mars 2023 17:34 - Robot Gitea

- Statut changé de Solution proposée à Solution validée

Emmanuel Cazenave (ecazenave) a approuvé une pull request sur Gitea concernant cette demande :

- URL : https://git.entrouvert.org/entrouvert/chrono/pulls/46

#21 - 14 mars 2023 17:47 - Robot Gitea

- Statut changé de Solution validée à Résolu (à déployer)

Benjamin Dauvergne (bdauvergne) a mergé une pull request sur Gitea concernant cette demande :

- URL : https://git.entrouvert.org/entrouvert/chrono/pulls/46
- Titre : misc: use zoneinfo instead of pytz for timezones (#71918)

- Modifications : https://git.entrouvert.org/entrouvert/chrono/pulls/46/files

#22 - 15 mars 2023 12:15 - Transition automatique

- Statut changé de Résolu (à déployer) à Solution déployée

#23 - 21 mai 2023 04:42 - Transition automatique

Automatic expiration