Authentic 2 - Development #37238

/api/users, filtre horaire lors du passage heure d'hiver

27 octobre 2019 11:34 - Frédéric Péters

Statut:	Fermé	**Début:**	27 octobre 2019
Priorité:	Normal	**Echéance:**	
Assigné à:	Benjamin Dauvergne	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

```
File "/usr/lib/python2.7/dist-packages/authentic2/api_views.py" in handle_exception
  92.             response = super(ExceptionHandlerMixin, self).handle_exception(exc)

File "/usr/lib/python2.7/dist-packages/rest_framework/views.py" in dispatch
  463.             response = handler(request, *args, **kwargs)

File "/usr/lib/python2.7/dist-packages/rest_framework/mixins.py" in list
  40.         queryset = self.filter_queryset(self.get_queryset())

File "/usr/lib/python2.7/dist-packages/rest_framework/generics.py" in filter_queryset
  151.             queryset = backend().filter_queryset(self.request, queryset, self)

File "/usr/lib/python2.7/dist-packages/rest_framework/filters.py" in filter_queryset
  119.             return filter_class(request.query_params, queryset=queryset).qs

File "/usr/lib/python2.7/dist-packages/django_filters/rest_framework/filterset.py" in qs
  50.             return super(FilterSet, self).qs

File "/usr/lib/python2.7/dist-packages/django_filters/filterset.py" in qs
  206.             if not self.form.is_valid():

File "/usr/lib/python2.7/dist-packages/django/forms/forms.py" in is_valid
  183.         return self.is_bound and not self.errors

File "/usr/lib/python2.7/dist-packages/django/forms/forms.py" in errors
  175.             self.full_clean()

File "/usr/lib/python2.7/dist-packages/django/forms/forms.py" in full_clean
  384.         self._clean_fields()

File "/usr/lib/python2.7/dist-packages/django/forms/forms.py" in _clean_fields
  402.                     value = field.clean(value)

File "/usr/lib/python2.7/dist-packages/django/forms/fields.py" in clean
  160.         value = self.to_python(value)

File "/usr/lib/python2.7/dist-packages/django/forms/fields.py" in to_python
  484.         result = super(DateTimeField, self).to_python(value)

File "/usr/lib/python2.7/dist-packages/django/forms/fields.py" in to_python
  405.                     return self.strptime(value, format)

File "/usr/lib/python2.7/dist-packages/django_filters/fields.py" in strptime
  123.             return handle_timezone(parsed)

File "/usr/lib/python2.7/dist-packages/django_filters/utils.py" in handle_timezone
  149.         return make_aware(value, timezone.get_current_timezone(), is_dst)

File "/usr/lib/python2.7/dist-packages/django_filters/compat.py" in make_aware
  69.         return make_aware_orig(value, timezone, is_dst)
```

```
File "/usr/lib/python2.7/dist-packages/django/utils/timezone.py" in make_aware
  285.         return timezone.localize(value, is_dst=is_dst)

File "/usr/lib/python2.7/dist-packages/pytz/tzinfo.py" in localize
  349.             raise AmbiguousTimeError(dt)

Exception Type: AmbiguousTimeError at /api/users/
Exception Value: 2019-10-27 02:58:07
Request information:
USER: CT-GNM-REC (70de91)

GET:
modified__gt = u'2019-10-27T02:58:07'
```

Révisions associées

Révision 173f63f6 - 12 novembre 2019 11:06 - Benjamin Dauvergne

api: work around ambiguous time error on DST change (#37238)

Historique

#1 - 29 octobre 2019 21:24 - Benjamin Dauvergne

C'est un bug très gênant et pour l'instant je préfère que ça plante parce que si c'est utilisé pour faire de la synchro le fait que la date soit mal interprété peut faire rater quasiment 1h de changements, idéalement pour accepter tout de même une date sans timezone il faudrait calculer la valeur avec is_dst=True et is_dst=False et garder la plus antérieure; mais ici tout est délégué à un champ de formulaire dans django-filter donc il faudrait pondre le notre.

#2 - 29 octobre 2019 23:31 - Benjamin Dauvergne

- Assigné à mis à Benjamin Dauvergne

#3 - 29 octobre 2019 23:33 - Benjamin Dauvergne

- Fichier 0001-api-work-around-ambiguous-time-error-on-DST-change-3.patch ajouté

- Tracker changé de Bug à Development

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

#4 - 31 octobre 2019 12:02 - Nicolas Roche

Oui, ça fonctionne parfaitement, mais qu'est-ce que ça fait mal à la tête !

3 remarques cependant :

- J'ai l'impression que ton test ne couvre pas complètement l'ambiguïté parce que tu aurais enregistrés les utilisateurs 1 heure après le changement d'heure.

Voici les tests que j'ai utilisés :

```
def test_filter_users_by_last_modification(app, admin, simple_user, freezer):
    app.authorization = ('Basic', (admin.username, admin.username))
    tz = pytz.timezone('Europe/Paris')
    # CET: Central Europe Time   vs CEST: Central Europe Summer Time)
    #  ^-- STD: standard time of tz   ^-- DST: daylight saving time
    #  ^-- UTC+1                      ^-- UTC+2

    #      spring            autumn
    # UTC  .....1h.....    .....1h....
    # CET  .....2h x           2h.x..
    # CEST    x 3h.....    ...x.3h
    #  2:30 do not exist    2:30 exists twice
    #  => NonExistent       => Ambiguous

    # 26 October 2019
    summer = datetime.datetime(2019, 10, 27, 1, 59, 59)
    ambiguous = datetime.datetime(2019, 10, 27, 2, 30, 0)
    winter = datetime.datetime(2019, 10, 27, 3, 0, 0)
    assert tz.dst(summer) == datetime.timedelta(0, 3600)  # CEST
    with pytest.raises(pytz.exceptions.AmbiguousTimeError):
        tz.dst(ambiguous)
```

```
    assert tz.dst(winter) == datetime.timedelta(0)  # CET

    # 29 March 2020
    winter = datetime.datetime(2020, 3, 29, 1, 59, 59)
    non_existent = datetime.datetime(2020, 3, 29, 2, 30, 0)
    summer = datetime.datetime(2020, 3, 29, 3, 0, 0)
    assert tz.dst(winter) == datetime.timedelta(0)  # CET
    with pytest.raises(pytz.exceptions.NonExistentTimeError):
        tz.dst(non_existent)
    assert tz.dst(summer) == datetime.timedelta(0, 3600)  # CEST

    # normal (20 February 2020)
    freezer.move_to('2020-02-20T01:00:00Z')  # UTC
    admin.save()
    simple_user.save()
    resp = app.get('/api/users/', params={'modified__lt': '2020-02-20T02:30:00'})
    assert len(resp.json['results']) == 2
    resp = app.get('/api/users/', params={'modified__gt': '2020-02-20T02:30:00'})
    assert len(resp.json['results']) == 0

    # AmbiguousTimeError (27 October 2019)
    freezer.move_to('2019-10-27T01:00:00Z')  # UTC
    admin.save()
    simple_user.save()
    resp = app.get('/api/users/', params={'modified__lt': '2019-10-27T02:30:00'})
    assert len(resp.json['results']) == 2
    resp = app.get('/api/users/', params={'modified__gt': '2019-10-27T02:30:00'})
    assert len(resp.json['results']) == 2

    # NonExistentTimeError (29 March 2020)
    freezer.move_to('2020-03-29T01:00:00Z')  # UTC
    admin.save()
    simple_user.save()
    resp = app.get('/api/users/', params={'modified__lt': '2020-03-29T02:30:00'})
    assert len(resp.json['results']) == 2
    resp = app.get('/api/users/', params={'modified__gt': '2020-03-29T02:30:00'})
    assert len(resp.json['results']) == 2
```

- Étendre la correction au passage à l'heure de printemps :
  il n'y a rien d'autre à faire que d'utiliser InvalidTimeError, qui regroupe les 2 exceptions AmbiguousTimeError et NonExistentTimeError.

- Pour moi il manque un commentaire sur la stratégie que tu as appliqué pour lever l'ambiguïté (sélection soit avare, soit gourmande). Et je serais d'avis de remplacer la magie du tri par une utilisation explicite du paramètre is_dst (daylight saving time).

```
  > [str(p) for p in possible]
  ['2019-10-27 02:30:00+02:00', '2019-10-27 02:30:00+01:00']
```

Par exemple :

```
# override to handle ambiguous naive DateTime on DST change
class IsoDateTimeField(IsoDateTimeField):
    def __init__(self, *args, **kwargs):
        self.is_dst = kwargs.pop('is_dst')
        assert isinstance(self.is_dst, bool)
        super(IsoDateTimeField, self).__init__(*args, **kwargs)

    def strptime(self, value, format):
        try:
            return super(IsoDateTimeField, self).strptime(value, format)
        except InvalidTimeError:
            parsed = parse_datetime(value)
            return handle_timezone(parsed, is_dst=self.is_dst)

class IsoDateTimeFilter(IsoDateTimeFilter):
    @property
    def field_class(self):
        # greedy selection to prevent InvalidTimeError
        if self.lookup_expr.startswith('gt'):
            # daylight saving time (summer)
            return partial(IsoDateTimeField, is_dst=True)
        elif self.lookup_expr.startswith('lt'):
            # standard time of the time zone (winter)
```

```
            return partial(IsoDateTimeField, is_dst=False)
        else:
            raise NotImplementedError
```

```
    def filter(self, qs, value):
        return super(IsoDateTimeFilter, self).filter(qs, value)
```

#5 - 31 octobre 2019 15:31 - Benjamin Dauvergne

On nous enverra jamais de date NonExistent, c'est pour cela que je ne le gère pas (dans l'idée que le client gère bien le changement d'heure juste il oublie de nous envoyer sa timezone) mais oui je vais changer InvalidTimeError si tu dis que ça suffit à cacher le souci.

Ensuite on s'en fout un peu parce que je ne sais même pas où c'est mais il y a des pays où l'heure d'hiver c'est +1 pas -1 (ouais c'est un gros bordel), donc je préférai mon idée de calculer les deux et de prendre l'heure la plus tôt ou la plus tard selon le filtre, je comprends mieux comme cela.

#6 - 31 octobre 2019 17:12 - Nicolas Roche

> prendre l'heure la plus tôt ou la plus tard selon le filtre

ce n'est toujours pas clair pour moi :
j'ai l'impression que le calcul de "l'heure la plus tôt" ce fait en fonction de la représentation des 2 dates comparées.
(ie : "CEST" < "CET" dans la liste triée possible alors que j'imaginai que +01:00 viendrait avant +02:00)

#7 - 31 octobre 2019 17:26 - Benjamin Dauvergne

Nicolas Roche a écrit :

> > prendre l'heure la plus tôt ou la plus tard selon le filtre
>
> ce n'est toujours pas clair pour moi :
> j'ai l'impression que le calcul de "l'heure la plus tôt" ce fait en fonction de la représentation des 2 dates comparées.
> (ie : "CEST" < "CET" dans la liste triée possible alors que j'imaginai que +01:00 viendrai avant +02:00)

Les comparaisons se font toujours en temps absolu (UTC si tu veux), sinon ça n'a pas de sens.

#8 - 01 novembre 2019 03:10 - Nicolas Roche

- Statut changé de Solution proposée à Solution validée

Désolé, j'ai moi même dessiné sur le schéma que pour 2 mêmes horaires, CEST vient avant CET.
En fait je viens de comprendre pourquoi UTC+2 vient avant UTC+1 :
c'est parce que c'est plus à l'Est (et que donc c'est avant dans le temps) !
Ouf, fini.

#9 - 01 novembre 2019 13:36 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Désolé, j'ai moi même dessiné sur le schéma que pour 2 même horaires, CEST vient avant CET.
> En fait je viens de comprendre pourquoi UTC+2 vient avant UTC+1 :
> c'est parce que c'est plus à l'Est (et que donc c'est avant dans le temps) !
> Ouf, fini.

Oui ou juste parce que pour revenir à UTC tu fais -2 heures pour le premier et -1 heure pour le second, donc en partant de la même heure le premier est avant le deuxième :)

#10 - 12 novembre 2019 11:25 - Benjamin Dauvergne

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

```
commit 173f63f647d0f6f61fcff36cc8669a12e7af304e
Author: Benjamin Dauvergne <bdauvergne@entrouvert.com>
Date:   Tue Oct 29 23:31:54 2019 +0100
```

```
    api: work around ambiguous time error on DST change (#37238)
```

#11 - 04 décembre 2019 18:15 - Frédéric Péters

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

Fichiers

0001-api-work-around-ambiguous-time-error-on-DST-change-3.patch	4,79 ko	29 octobre 2019	Benjamin Dauvergne