Authentic 2 - Bug #29531

export CSV des utilisateurs vs nombreux utilisateurs

07 janvier 2019 15:55 - Frédéric Péters

Statut:	Fermé	**Début:**	07 janvier 2019
Priorité:	Normal	**Echéance:**	
Assigné à:	Emmanuel Cazenave	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Ça peut prendre du temps et dépasser le timeout de connexion HTTP; doit-on étendre celui-ci, ou imagine-t-on de manière plus sophistiquée la possibilité d'une génération "asynchrone" de l'export ?

Révisions associées

Révision 56c72c2c - 15 janvier 2019 16:11 - Emmanuel Cazenave

use directly tablib instead of django-export-export (#29531)

Révision 93c52a94 - 15 janvier 2019 16:11 - Emmanuel Cazenave

custom attributes prefetching (#29531)

Historique

#1 - 08 janvier 2019 16:15 - Emmanuel Cazenave

- Assigné à mis à Emmanuel Cazenave

A la demande d'un CPT dont je tairais le nom, je vais essayer de m'y coller sauf si quelqu'un d'autre est chaud.

Et si a quelqu'un a déjà des billes sur ce sujet que je découvre ... l'intitulé suggère qu'une recherche d'optimisation synchrone n'est pas possible, et au doigt mouillé j'imagine que c'est sur les attributs utilisateurs que ça prend un temps incompressible.

#3 - 08 janvier 2019 16:25 - Emmanuel Cazenave

Emmanuel Cazenave a écrit :

> Et si a quelqu'un a déjà des billes sur ce sujet que je découvre ... l'intitulé suggère qu'une recherche d'optimisation synchrone n'est pas possible, et au doigt mouillé j'imagine que c'est sur les attributs utilisateurs que ça prend un temps incompressible.

Si j'ai bon et vu le besoin exprimé dans #29526, j'imagine aussi la possibilité d'un export csv simplifié qui zapperait les attributs.

#4 - 08 janvier 2019 16:28 - Frédéric Péters

> une recherche d'optimisation synchrone n'est pas possible,

Si si, tu peux explorer ça aussi.

#5 - 14 janvier 2019 13:35 - Benjamin Dauvergne

On peut vraiment améliorer les performances synchrones, parce que sur la base strasbourg:

- un list(User.objects.all()) prend 219 ms en local
- un list(AttributeValue.objects.all()) prend 1,21 s en local
- mais un list(User.objects.prefetch_related('attribute_values')) prend 14 s en local
- un list(AttributeValue.objects.select_related('attribute')) prend 4,1 s
- et un list(User.objects.prefetch_related('attribute_values__attribute')) prend dans les 10s

En optimisant le mapping des AttributeValue vers les User on devrait gagner pas mal (il faut se passer de django-import-export au passage).

#6 - 14 janvier 2019 14:10 - Emmanuel Cazenave

Benjamin Dauvergne a écrit :

> En optimisant le mapping des AttributeValue vers les User on devrait gagner pas mal (il faut se passer de django-import-export au passage).

Je ne comprends pas, à quoi tu penses ? Qu'est ce qu'on peut optimiser dans une GenericRelation ?

#7 - 14 janvier 2019 15:12 - Benjamin Dauvergne

```
users = OrderedDict()
for user in User.objects.all():
    users[user.pk] = (user, {})

at_mapping = {a.id: a for a in Attribute.objects.all()}
for atv in AttributeValue.objects.filter(content_type=ContentType.object.get_for_model(User)):
     at = at_mapping[atv.attribute_id]
     users[atv.object_id][1][at.name] = atv.content
```

C'est ce que fait preftech_related() mais moins bien visiblement.

#8 - 14 janvier 2019 18:06 - Emmanuel Cazenave

- Statut changé de Nouveau à En cours

Pas analysé pourquoi mais "se passer de django-import-export", ça divise déjà le temps par deux, je suis en train de faire ça.

#9 - 15 janvier 2019 15:08 - Emmanuel Cazenave

- Fichier 0001-use-directly-tablib-instead-of-django-export-export-.patch ajouté

- Fichier 0002-custom-attributes-prefetching-29531.patch ajouté

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

0001 est un peu hackish pour éviter la duplication avec ExportMixin.
0002 prefetch manuel des attributs (il reste du prefetch django des OU et des rôles planqué).

Bilan des courses sur les deux patchs, un jeu de tests de 5000 utilisateurs, 10 attributs,
on passe grosso modo de 15 secondes à 4 secondes.

#10 - 15 janvier 2019 15:12 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Ça m'a l'air bien, la limite reviendra mais plutôt vers 50 000 utilisateurs que 20 000 (on y est déjà à plusieurs endroits je pense, GL par exemple).

#11 - 15 janvier 2019 16:18 - Emmanuel Cazenave

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

```
commit 56c72c2cf2fdff2e6a9f018d5554a51ecc815a80
Author: Emmanuel Cazenave <ecazenave@entrouvert.com>
Date:   Tue Jan 15 10:43:05 2019 +0100

    use directly tablib instead of django-export-export (#29531)

commit 93c52a940cff36778602246d88129e2dc3a9b7e5
Author: Emmanuel Cazenave <ecazenave@entrouvert.com>
Date:   Tue Jan 15 14:52:06 2019 +0100

    custom attributes prefetching (#29531)
```

#12 - 17 janvier 2019 13:15 - Frédéric Péters

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

Fichiers

0002-custom-attributes-prefetching-29531.patch	3,86 ko	15 janvier 2019	Emmanuel Cazenave
0001-use-directly-tablib-instead-of-django-export-export-.patch	3,87 ko	15 janvier 2019	Emmanuel Cazenave