Authentic 2 - Bug #53151

Timeout sur /manage/users/xx/roles/ (quand on a 18 000 rôles)

15 avril 2021 11:48 - Emmanuel Cazenave

Statut:	Fermé	**Début:**	15 avril 2021
Priorité:	Normal	**Echéance:**	
Assigné à:	Emmanuel Cazenave	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Pourtant il y a de la pagination, mais quand même plouf timeout.

Demandes liées:

Lié à Authentic 2 - Development #53170: Compatibilité avec django-tables 2.1.1	**Nouveau**	**15 avril 2021**
Copié depuis Authentic 2 - Development #52854: Timeout sur /manage/users/xx/...	**Rejeté**	**08 avril 2021**

Révisions associées

Révision f2cdd451 - 17 mai 2021 15:11 - Emmanuel Cazenave

manager: paginate user role view (#53151)

Historique

#1 - 15 avril 2021 12:24 - Valentin Deniaud

C'est peut-être quand même cette histoire de pagination qui est en cause, « The default Paginator wants to count the number of items, which might be an expensive operation for large QuerySets . In those cases, you can use LazyPaginator » https://django-tables2.readthedocs.io/en/latest/pages/pagination.html?highlight=singletableview#lazy-pagination.

#2 - 15 avril 2021 12:30 - Benjamin Dauvergne

Je pense que c'est ça, parce que la requête en dessous est complexe et le count() doit demander un scan complet.

PS: j'aurai bien répondu avec un screenshot de l'écran debug-toolbar SQL, mais là ça ne marche plus dans mon publik-devinst, je regarde pourquoi.

#3 - 15 avril 2021 12:37 - Benjamin Dauvergne

LazyPaginator n'a pas l'air disponible dans la version de django-tables2 qu'on utilise en prod (1.21.2), c'est introduit en 2.x :

```
$ grep LazyPa `dpkg -L python3-django-tables2 | grep py$`
$
```

#4 - 15 avril 2021 12:50 - Benjamin Dauvergne

- Copié depuis Development #52854: Timeout sur /manage/users/xx/roles/ (quand on a 18 000 rôles) ajouté

#5 - 15 avril 2021 15:40 - Emmanuel Cazenave

- Lié à Development #53170: Compatibilité avec django-tables 2.1.1 ajouté

#6 - 21 avril 2021 17:30 - Emmanuel Cazenave

Emmanuel Cazenave a écrit :

> ~~Pourtant il y a de la pagination~~

Mais en fait non, c'est désactivé (ne jamais croire sur parole ses collègues) :

```
class UserRolesView(HideOUColumnMixin, BaseSubTableView):

....

    @property
    def table_pagination(self):
```

```
        if self.is_ou_specified():
            return False
        return None
```

Mais même en activant la pagination ça timeout quand même .... retour à l'analyse sans pagination pour ne pas tout mélanger.

Tentative de mesurer les choses via la debugtoolbar qui s'enlise complètement dans du code récursif de django/template/base.py.

J'essaie avec mon couteau et dieu sait quoi, trafiquer OuUserRolesTable et voir ce que ça change. Il semble la colonne **via** soit très coûteuse à afficher :

```
    via = tables.TemplateColumn(
        '''{% for rel in record.via %}{{ rel.child }} {% if not forloop.last %}, {% endif %}{% endfor %}''',
        verbose_name=_('Inherited from'),
        orderable=False,
    )
```

Ce qui amène à s'intéresser au queryset qui est utilisé par la table :

```
qs = qs.prefetch_related(models.Prefetch('child_relation', queryset=rp_qs, to_attr='via'))
```

Mais je ne reproduis pas du tout les lenteurs en évaluant le queryset en dehors d'un template .... plouf l'impasse, j'en ai marre de pas comprendre.

#7 - 22 avril 2021 07:49 - Benjamin Dauvergne

Il faut juste regarder les requêtes SQL émises dans la debug-toolbar, tu devrais en voir une plus longue que les autres (si c'est bien une lenteur due à une requête SQL).

#8 - 22 avril 2021 09:46 - Emmanuel Cazenave

Benjamin Dauvergne a écrit :

> Il faut juste regarder les requêtes SQL émises dans la debug-toolbar, tu devrais en voir une plus longue que les autres (si c'est bien une lenteur due à une requête SQL).

J'aurais du préciser mais non c'est pas du tout ça, temps global sur toutes les requêtes un peu long mais marginal par rapport au temps global.

#9 - 22 avril 2021 10:50 - Benjamin Dauvergne

Si tu as les instants de début des requêtes SQL et de fin de la requête Django on doit pouvoir dire si c'est le temps perdu est entre des requêtes ou après que toutes les requêtes aient été faite ? Ça pourrait pointer le bout de code python qui traite les résultats et est un peu lent.

Aussi il me semble que django-tables2 utilise mal la pagination SQL (i.e. il fait list(qs)[debut:fin] au lieu de qs[debug:fin]), il faudrait vérifier qu'il ne récupère pas tous les résultats ça se voir dans le SQL si tu vois des LIMIT/OFFSET ou pas; en fait je ne sais pas trop si le temps rapporté pour une requête SQL est celui de la récupération totale des résultats ou seulement de la requête initiale avant récupération des lignes (ça peut être trompeur) via cursor.fetchmany().

#11 - 06 mai 2021 14:12 - Emmanuel Cazenave

- Statut changé de Nouveau à En cours

- Assigné à mis à Emmanuel Cazenave

#12 - 06 mai 2021 17:04 - Emmanuel Cazenave

- Fichier 0001-manager-speed-up-user-role-view-53151.patch ajouté

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

Les templates dans dans les templates c'est pas bon pour la perf apparemment. Avec 5000 rôles je passe de 28 à 6 secondes avec ce patch.

#13 - 06 mai 2021 17:20 - Emmanuel Cazenave

- Fichier 0001-manager-speed-up-user-role-view-53151.patch ajouté

Un espace en moins dans un test.

#14 - 06 mai 2021 17:35 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Si ça te paraît suffisant (mais ça m'intéresse de savoir comment ça peut bouffer 22 secondes de plus quand c'est paginé et que donc ça ne s'exécute

que quelques dizaines de fois).

#15 - 10 mai 2021 14:05 - Emmanuel Cazenave

- Statut changé de Solution validée à En cours

Ça n'est pas paginé, d'où le gain. Mais en testant sur 20000 rôles et pas 5000, ça remonte à 60 secondes, inutilisable.

Je vais regarder pour la pagination.

#16 - 10 mai 2021 14:48 - Emmanuel Cazenave

- Fichier 0001-manager-paginate-user-role-view-53151.patch ajouté

- Statut changé de En cours à Solution proposée

Dans la série faire beaucoup de bruit pour rien, j'étais persuadé d'avoir testé la pagination et constaté que ça ramait encore à fond. Mais là je teste à nouveau avec ce bête patch sur 20000 rôles et ça répond dans des temps tout à fait acceptable. J'ai du m'emmêler les pinceaux la première fois.

#17 - 10 mai 2021 15:35 - Benjamin Dauvergne

Emmanuel Cazenave a écrit :

> Dans la série faire beaucoup de bruit pour rien, j'étais persuadé d'avoir testé la pagination et constaté que ça ramait encore à fond. Mais là je teste à nouveau avec ce bête patch sur 20000 rôles et ça répond dans des temps tout à fait acceptable. J'ai du m'emmêler les pinceaux la première fois.

Est-ce que ça marche aussi simplement en enlevant la méthode ? Parce que c'est bizarre de passer de False/None à True.

#18 - 10 mai 2021 15:55 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Je valide quand même ma remarque est un détail.

#19 - 10 mai 2021 15:57 - Emmanuel Cazenave

Ça passe aussi sans la méthode, je pousserai cette nouvelle version.

#20 - 17 mai 2021 15:12 - Emmanuel Cazenave

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

```
commit f2cdd451dad7c9ed2430a7c99483da9aef29fae6
Author: Emmanuel Cazenave <ecazenave@entrouvert.com>
Date:   Mon May 10 14:22:14 2021 +0200

    manager: paginate user role view (#53151)
```

#21 - 18 mai 2021 20:16 - Frédéric Péters

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

Fichiers

0001-manager-speed-up-user-role-view-53151.patch	2,92 ko	06 mai 2021	Emmanuel Cazenave
0001-manager-speed-up-user-role-view-53151.patch	3,38 ko	06 mai 2021	Emmanuel Cazenave
0001-manager-paginate-user-role-view-53151.patch	816 octets	10 mai 2021	Emmanuel Cazenave