Combo - Development #17645

performance : liste des cellules

16 juillet 2017 14:15 - Frédéric Péters

Statut:	Rejeté	**Début:**	16 juillet 2017
Priorité:	Normal	**Echéance:**	
Assigné à:	Frédéric Péters	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	

Description

Pour le moment on ne le remarque pas encore trop sur notre infra (d'où priorité: bas) mais le rendu d'une page exige un nombre beaucoup trop élevé de requêtes.

Pour trouver les cellules à afficher sur une page, il faut autant d'appels que de type de cellule, et ensuite, il faut multipler ça par la position de la page dans la hiérarchie (pour trouver le contenu des cellules "identique au parent").

Déjà, si au niveau d'une page on gardait un cache du type de cellules qui s'y trouvent, on réduirait drastiquement le nombre de requêtes.

Historique

#1 - 07 janvier 2018 15:49 - Frédéric Péters

- Statut changé de Nouveau à En cours

- Assigné à mis à Frédéric Péters

- Priorité changé de Bas à Normal

Transformation de CellBase en modèle concret (abstract = False) avec un tas de migrations dans wip/concrete-cellbase-17645. (déroulé inspiré de http://tech.agilitynerd.com/django-migrate-abstract-concrete-base-class.html, des différences pour gérer SQLite et pour le moment la conservation des attributs subclass/subclass_id de transition).

Ça tourne sur mon installation locale; des tests échouent pour les migrations (parce que la sérialisation json ne reprend plus toutes les infos) et sur un tests de tableau de bord (sans doute pour une raison similaire).

Ça amène un tas de migrations du coup je vais tâcher de finir ça rapidement et en attendant je vais sans doute hésiter avant d'accepter de nouvelles migrations.

Ensuite, pour parler performances, il y aura à modifier CellBase.get_cells() pour ne plus passer sur toutes les classes de cellule.

#2 - 08 janvier 2018 12:47 - Frédéric Péters

> Ça tourne sur mon installation locale; des tests échouent pour les migrations (parce que la sérialisation json ne reprend plus toutes les infos) et sur un tests de tableau de bord (sans doute pour une raison similaire).

Et c'est désormais adapté (mais ça a demandé de répliquer pas mal du code de sérialisation, pas idéalement foutu côté django).

#3 - 08 janvier 2018 16:35 - Frédéric Péters

Nouveau commit dans la branche qui ajoute django-model-utils; les tests passent et à titre indicatif, en prenant test_page_footer_acquisition :

```
    page = Page(title='Home', slug='index', template_name='standard')
    page.save()
    cell = TextCell(page=page, placeholder='footer', text='BARFOO', order=0)
    cell.save()

    page = Page(title='Second', slug='second', template_name='standard')
    page.save()
    ParentContentCell(page=page, placeholder='footer', order=0).save()
    resp = app.get('/second/', status=200)
```

Ce app.get('/second/') faisait 83 requêtes, en fait désormais 8.

#4 - 08 janvier 2018 19:46 - Frédéric Péters

Poussé quelques adaptations supplémentaires, migrations nécessaires pour deux endroits qui pointaient vers des cellules (le manytomany avec les couches d'une carte et les tuiles d'un tableau de bord).

Aussi fait des tests sur une page type "services" gnm (ma copie locale), ça passe de 128 à 53 requêtes MAIS le temps de rendu passe de 285ms à 646ms. (la requête de django-model-utils avec une tonne de JOIN pour récupérer toutes les classes enfant semble super couteuse).

#5 - 08 janvier 2018 22:43 - Frédéric Péters

> passe de 285ms à 646ms

En grande partie c'était amené à cause des badges; dans la version précédente seules les cellules pouvant porter des badges étaient interrogées et lors de la conversion c'était passé à interroger tout le monde; c'est corrigé dans un nouveau commit ("perf: optimize badge checking").

À tracer la suite, l'élément suivant qui prend du temps c'est extend_with_parent_cells (surtout quand il y a une hiérarchie de pages de quelques niveaux); j'ai tapé un autre commit pour optimiser ça en tapant moins dans la db. (mais ce n'est pas lié au passage de modèle abstrait à concret, ça pourrait aller dans master aussi).

Avec ça, ma page de test passe à 239ms et le temps passé est désormais majoritairement dans le render() de la page.

#6 - 09 janvier 2018 10:24 - Frédéric Péters

- Statut changé de En cours à Rejeté

Donc, l'enseignement, c'est qu'on ne gagne rien à chercher de ce côté, on a trop peu d'endroits dans combo où on peut se contenter des informations de base d'une cellule.

Vu le prix à payer (migrations SQL écrites à la main, sérialisation "custom", dépendance à django-model-utils), j'abandonne l'idée ici.