Passerelle - Development #11260

mise en base des données du CSV

07 juin 2016 19:10 - Frédéric Péters

Statut:	Fermé	**Début:**	07 juin 2016
Priorité:	Bas	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Aujourd'hui on charge à la volée le fichier CSV, ça marche pour une liste des cinq lycées de la commune, ça ne va pas être top sur des beaucoup plus grosses listes (genre la liste des rues du département); ce serait super si on pouvait charger en base le CSV.

Demandes liées:

Lié à Passerelle - Bug #14404: connecteur tableur : perfs déplorables sur un ...	**Nouveau**	**21 décembre 2016**

Révisions associées

Révision f3ad7a03 - 05 septembre 2018 16:14 - Frédéric Péters

csv: use dictionaries to manipulate rows (#11260)

Révision 07aafc55 - 05 septembre 2018 16:14 - Frédéric Péters

csv: keep file data in database (#11260)

Révision 5daa2030 - 05 septembre 2018 16:37 - Frédéric Péters

csv: add missing migration (#11260)

Historique

#1 - 21 décembre 2016 11:34 - Frédéric Péters

- Lié à Bug #14404: connecteur tableur : perfs déplorables sur un fichier ods ajouté

#2 - 21 décembre 2016 11:35 - Jean-Baptiste Jaillet

- Assigné à mis à Jean-Baptiste Jaillet

#3 - 22 décembre 2016 01:33 - Jean-Baptiste Jaillet

- Fichier 0001-csvdatasource-save-rows-values-in-database-11260.patch ajouté

- Patch proposed changé de Non à Oui

pyexcel-xls c'est vraiment de la merde...
Bref, contrairement à son cousin ods, il ne peut faire get_data qu'avec le file path (parce que le read ou l'objet file non...). Seulement voilà, quand on upload un fichier, c'est un InMemoryFile, donc le fichier n'existe pas.
J'ai dû jouer avec les signaux et c'est la manière la plus élégante que j'ai trouvée.
Pour les tests ma foi, je ne vois pas de test spécifique pour l'ajout de ce champs, j'ai en revanche fait tourner tous les tests de csvdatasource et quelle a été mon euphorie quand enfin j'ai vu s'afficher ce doux "104 passed". Le bonheur est fait de petites victoires.

#4 - 22 décembre 2016 08:46 - Frédéric Péters

```
rows_values = jsonfield.JSONField(editable=False)
```

Il semble que j'aurais du développer pourquoi "ça ne va pas être top sur des beaucoup plus grosses listes". Une des raisons c'est se trouver devoir tout charger en mémoire. Et ici, donc, ça ne change rien, c'est juste que plutôt que sortir les dizaines ou centaines de Mo d'un fichier ils vont sortir de la db. (et j'aime beaucoup postgresql mais pour autant je n'y taperais pas des centaines de Mo dans un champ texte).

#5 - 22 décembre 2016 12:04 - Jean-Baptiste Jaillet

- Statut changé de Nouveau à En cours

Hum.. Ok, je pensais que les accès en base (même si j'ai conscience que c'est long à partir d'un certain volume) étaient plus rapides.
Et si le fichier est mal fait, on aura en base que les valeurs pleines.

Du coup si ce n'est pas stocké comme ça je veux bien une idée je ne vois comment faire autrement pour garder les données en base.

#6 - 22 décembre 2016 12:06 - Thomas Noël

Rien à voir avec le stockage en base, mais https://dev.entrouvert.org/issues/14404#note-5

#7 - 22 décembre 2016 12:24 - Frédéric Péters

- Priorité changé de Normal à Bas

- Patch proposed changé de Oui à Non

> Du coup si ce n'est pas stocké comme ça je veux bien une idée je ne vois comment faire autrement pour garder les données en base.

Ok, j'ai diminué la priorité du ticket, il y a vraiment aucune urgence ici et oui la réponse peut être compliquée. Il y a mille possibilités qui peuvent être imaginées, en partant de l'exemple donné "la liste des rues du département", on peut se souvenir de ce qui a été fait dans le connecteur base adresse, commencer par avoir un objet par ligne, continuer en se disant qu'on peut considérer un max de n colonnes signifiantes et leur donner des champs dédiés, le reste de la ligne pouvant aller dans un jsonfield; trouver alors que cet objet peu typé va pas être terrible et chercher à créer une représentation des données qui soit souple et performante, passer par l'idée de simplement générer une db sqlite sur le côté, etc.

#8 - 22 décembre 2016 12:28 - Frédéric Péters

Et on peut aussi vraiment se dire que l'exemple donné a été résolu de manière spécifique, qu'à partir d'un certain volume de données ça devrait être le cas, que le connecteur tableur ne peut pas tout accueillir.

#9 - 22 décembre 2016 13:41 - Thomas Noël

Frédéric Péters a écrit :

> qu'à partir d'un certain volume de données ça devrait être le cas, que le connecteur tableur ne peut pas tout accueillir.

et qu'inversement, que pour des petites listes de quelques lignes et quelques colonnes, gérer ça via un CSV, c'est chiant.

#10 - 02 octobre 2017 23:41 - Frédéric Péters

- Assigné à Jean-Baptiste Jaillet supprimé

#11 - 14 août 2018 14:04 - Frédéric Péters

- Fichier 0001-csv-keep-file-data-in-database-11260.patch ajouté

- Statut changé de En cours à Solution proposée

- Patch proposed changé de Non à Oui

De manière très basique, au chargement ça enregistre les lignes en db (et uniquement les colonnes demandées); c'est surtout motivé par les fichiers ods qui sont trop lents à charger (cf #14404).

#12 - 14 août 2018 15:37 - Frédéric Péters

- Fichier 0001-csv-keep-file-data-in-database-11260.patch ajouté

Correction pour forcer la création d'une liste à partir de l'itérateur.

#13 - 03 septembre 2018 23:26 - Benjamin Dauvergne

- séparer le début du patch index -> key dans son propre commit
- il me semble que si le CSV est vide ça va boucler infiniment:

```
    def get_cached_rows(self):
        found = False
        for row in TableRow.objects.filter(resource=self):
            found = True
            yield row.data
        if not found:
            # if there was no row probably the data was not cached in database
```

```
            # yet.
            self.cache_data()
            for data in self.get_cached_rows():
                yield data
```

(J'ai vérifié, un fichier vide donne bien 0 ligne de CSV:

```
In [3]: list(csv.reader(StringIO.StringIO('')))
Out[3]: []
```

)

- anecdotique, le champ line_number me semble inutile, dans le cas d'une création dans une unique transaction on est certain que id sera séquentiellement alloué, donc order by id suffirait.

Sur le fond je vois ça comme un pas vers l'application des critères directement aux lignes du modèle (via le type JSON natif de postgres) mis en cache, sinon je ne vois pas bien le progrès par rapport à CSV (il faudrait mesurer mais j'ai l'impression que ça n'ira pas tellement plus vite) mais ça doit éventuellement améliorer les choses pour des gros fichiers ODT ou XLS (il me semble que c'était un des cas d'usage initial, mais on aurait pu dans ce cas mettre en cache une version CSV sur disque).

#14 - 04 septembre 2018 09:03 - Frédéric Péters

séparer le début du patch index -> key dans son propre commit

Voilà, 0001.

il me semble que si le CSV est vide ça va boucler infiniment:

En effet; modulo qu'un CSV vide va être refusé parce qu'il n'arrivera pas à en déterminer le délimiteur de colonnes. Mais j'ai pu faire un test en mettant un ods vide pour valider la correction.

> anecdotique, le champ line_number me semble inutile, dans le cas d'une création dans une unique transaction on est certain que id sera séquentiellement alloué, donc order by id suffirait.

Je le trouve utile pour à un moment pouvoir aider au debug, permettre de pointer l'usager vers la ligne de son fichier.

> Sur le fond je vois ça comme un pas vers l'application des critères directement aux lignes du modèle (via le type JSON natif de postgres) mis en cache, sinon je ne vois pas bien le progrès par rapport à CSV (il faudrait mesurer mais j'ai l'impression que ça n'ira pas tellement plus vite) mais ça doit éventuellement améliorer les choses pour des gros fichiers ODT ou XLS (il me semble que c'était un des cas d'usage initial, mais on aurait pu dans ce cas mettre en cache une version CSV sur disque).

La motivation présente est bien d'éviter les bugs de lecture d'ods (c'est noté dans le commentaire au-dessus). Mais à un moment pouvoir profiter des requêtes JSON, tout à fait. (même si j'ai en parallèle l'idée que la partie "requêtes" pourrait être sortie de ce connecteur particulier, pouvoir s'appliquer à tout autre connecteur fournissant l'API "données", et qu'il n'y aura alors pas nécessairement de db en backend).

#15 - 04 septembre 2018 09:03 - Frédéric Péters

- Fichier 0002-csv-keep-file-data-in-database-11260.patch ajouté

- Fichier 0001-csv-use-dictionaries-to-manipulate-rows-11260.patch ajouté

#16 - 05 septembre 2018 15:19 - Benjamin Dauvergne

- Statut changé de Solution proposée à Solution validée

Ack.

#17 - 05 septembre 2018 16:15 - Frédéric Péters

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

```
commit 07aafc554307f008d7216d8821b3986207231113
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Tue Sep 4 08:11:35 2018 +0200

    csv: keep file data in database (#11260)

commit f3ad7a0347f6e0579408c6ef21b178da45f8b3ae
Author: Frédéric Péters <fpeters@entrouvert.com>
```

```
Date:   Tue Sep 4 08:10:03 2018 +0200

    csv: use dictionaries to manipulate rows (#11260)
```

#18 - 03 décembre 2018 15:14 - Benjamin Dauvergne

- Statut changé de Résolu (à déployer) à Fermé

Fichiers

0001-csvdatasource-save-rows-values-in-database-11260.patch	6,55 ko	22 décembre 2016	Jean-Baptiste Jaillet
0001-csv-keep-file-data-in-database-11260.patch	10,3 ko	14 août 2018	Frédéric Péters
0001-csv-keep-file-data-in-database-11260.patch	10,7 ko	14 août 2018	Frédéric Péters
0001-csv-use-dictionaries-to-manipulate-rows-11260.patch	6,53 ko	04 septembre 2018	Frédéric Péters
0002-csv-keep-file-data-in-database-11260.patch	13,5 ko	04 septembre 2018	Frédéric Péters