# Authentic 2 - Development #29193

## support JSONField 1.11 et django-jsonfield

18 décembre 2018 23:36 - Benjamin Dauvergne

| | | | |
|---|---|---|---|
| **Statut:** | Fermé | **Début:** | 18 décembre 2018 |
| **Priorité:** | Normal | **Echéance:** | |
| **Assigné à:** | Benjamin Dauvergne | **% réalisé:** | 100% |
| **Catégorie:** | | **Temps estimé:** | 0:00 heure |
| **Version cible:** | | | |
| **Patch proposed:** | Oui | **Planning:** | |

**Description**

Il faut pouvoir gérer la migration progressive vers le champs natif sans perdre la compatibilité sqlite et Django < 1.11 (pour l'instant).

### Révisions associées

**Révision d730dba5 - 18 janvier 2019 19:30 - Benjamin Dauvergne**

add compatibility layer for support of Django native JSONField (fixes #29193)

**Révision 000f6836 - 19 janvier 2019 10:36 - Benjamin Dauvergne**

compat: handle case of Django 1.11 without psycopg2 (fixes #29193)

**Révision cef04caf - 21 novembre 2019 17:02 - Paul Marillonnet**

Revert "add compatibility layer for support of Django native JSONField (fixes #29193)"

This reverts commit d730dba525215d615fa2b974d44183d16909f4d2.

### Historique

**#1 - 19 décembre 2018 08:59 - Benjamin Dauvergne**

*- Fichier 0001-add-compatibility-layer-for-support-of-Django-native.patch ajouté*

*- Patch proposed changé de Non à Oui*

Un poil acrobatique mais tous les tests passent en Django 1.11 et 1.8, j'ai pu vérifier localement que je peux passer de 1.8 à 1.11, changer manuellement le type de la colonne jwkset sur OIDCProvider de "jsonb" vers "text" et avoir au niveau du post_migrate une correction automatique (retour vers jsonb dans la mesure où la version de postgres le supporte).

Ça rend impossible de faire du multidb vers des dbs de types différents (genre default=postgresql et other=sqlite, comme je décide de l'implémentation du champ via un check sur django.db.connection qui est toujours la connection par défaut).

Le gros du code tordu c'est pour que ça n'impacte pas la création de migrations, qui font un usage un peu excessif de isinstance().

Concernant l'utilisation des fonctionnalités avancées de django.contrib.postgresql.fields.JSONField, si on veut garder toujours une compatibilité sqlite3 (et dans une moindre mesure Django < 1.11) il faut systématiquement les protéger avec un if authentic2.compat.use_django_native_field():.

**#2 - 19 décembre 2018 16:06 - Benjamin Dauvergne**

Dans le même genre, https://github.com/raphaelm/django-jsonfallback/, mais c'est plutôt dans une optique de support post Django 1.11 notamment pour mysql et sqlite.

**#3 - 19 décembre 2018 18:39 - Paul Marillonnet**

Une première relecture, avant d'appliquer le patch et de le tester en vrai :

Dans authentic2/apps.py :

Je pense qu'on peut laisser expected_type en minuscules
et donc retirer le upper() avant l'assignation à current_type.

J'ai l'impression que django.db.models.fields.Field.get_attname_column ne remonte par d'erreur si la colonne n'est pas trouvée, à la place il renvoie le nom d'attribut.
Je serais donc vigilant lorsqu'on fait un
cursor.fetchone()[0].upper()

Dans authentic2/compat.py :

Dans le constructeur de authentic2.compat.JSONField :

```
        except ImportError:
            raise
            pass
```

→ je crois que le raise est de trop, non ? ou alors assert self.__dj11_field à la place ?

la méthode contribute_to_class doit supporter un paramètre private_only (et, respecter l'interface, son équivalent déprécié virtual_only).

D'ailleurs cette méthode, dans le code django de contrib postgres se termine par un bout de code qui me paraît nécessaire aussi ici :

```
        if self.column:
            # Don't override classmethods with the descriptor. This means that
            # if you have a classmethod and a field with the same name, then
            # such fields can't be deferred (we don't have a check for this).
            if not getattr(cls, self.attname, None):
                setattr(cls, self.attname, DeferredAttribute(self.attname, cls))
```

Dans la méthode compat.JSONField.deconstruct :
Je ne comprends ce bout de code :

```
        if inspect.getframeinfo(previous_frame)[2] in ('serialize', 'deep_deconstruct'):
            d = d[1:]
```

Le code originel du Field django renvoie toujours un quadruplet dont le premier élément est le nom du champ. Pourquoi le retirer ici ?

Il faudra aussi m'expliquer d'où vienne jsonfield.fields.configure_database_connection et jsonfield.fields.connection_created.
Je ne trouve pas le code, et donc je ne peux pas comprendre ce que tu cherches à faire avec ces quelques lignes à la fin de compat.py.

Dans tests/settings.py :

```
+         'NAME': 'postgres',
```

Wut ?!

#### #4 - 19 décembre 2018 20:28 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Une première relecture, avant d'appliquer le patch et de le tester en vrai :
>
> Dans authentic2/apps.py :
>
> Je pense qu'on peut laisser expected_type en minuscules
> et donc retirer le upper() avant l'assignation à current_type.

C'est copié/collé mais pourquoi pas.

> J'ai l'impression que django.db.models.fields.Field.get_attname_column ne remonte par d'erreur si la colonne n'est pas trouvée, à la place il renvoie le nom d'attribut.

Hmm ? Je ne comprends pas, tu veux qu'il retourne quoi ?

> Je serais donc vigilant lorsqu'on fait un
> cursor.fetchone()[0].upper()
>
> Dans authentic2/compat.py :
>
> Dans le constructeur de authentic2.compat.JSONField :
>
> [...]
> → je crois que le raise est de trop, non ? ou alors assert self.__dj11_field à la place ?

Yep debug qui traîne.

> la méthode contribute_to_class doit supporter un paramètre private_only (et, respecter l'interface, son équivalent déprécié virtual_only).

C'est pour cela que j'ai mis kwargs, pour ignorer ça, mais je peux mettre un assert not kwargs pour être sûr que ça n'arrive jamais.

> D'ailleurs cette méthode, dans le code django de contrib postgres se termine par un bout de code qui me paraît nécessaire aussi ici :
> [...]

Pareil on va dire que ça ne doit pas arriver (définition d'un nom explicite de colonne impliquant la création d'un attribut déferré).

> Dans la méthode compat.JSONField.deconstruct :
> Je ne comprends ce bout de code :
> [...]
> Le code originel du Field django renvoie toujours un quadruplet dont le premier élément est le nom du champ. Pourquoi le retirer ici ?

Parce que sinon ça ne marche pas, voir le code de serialize et deep_deconstruct dans django.db.migrations.autodetector et autour; la méthode deconstruct des Field renvoie un quadruplet mais tous les autres objets des triplets; comme on hérite pas de Field le moteur de migration prend le champ pour un autre type d'objet.

> Il faudra aussi m'expliquer d'où vienne jsonfield.fields.configure_database_connection et jsonfield.fields.connection_created.
> Je ne trouve pas le code, et donc je ne peux pas comprendre ce que tu cherches à faire avec ces quelques lignes à la fin de compat.py.

Contourner le bug d'incompatibilité entre django.contrib.postgres.fiels.JSONField et jsonfield.JSONField, celui que tu as pointé.

> Dans tests/settings.py :
>
> [...]
>
> Wut ?!

Bug tordu, le nouveau champ fait un accès à la base à l'initialisation, et donc en dehors des tests, (pour connaître la version de postgres, mais dès que tu accèdes à django.db.connection même pour lire vendor qui n'est qu'une info coté Django, ça ouvre une connection), comme les settings des tests ne définissait pas NAME ça ne passait pas; l'autre possibilité ce serait de ne pas faire cette accès si DATABASES n'est pas initialisé correctement (il l'est ensuite dans les tests). Et j'ai mis postgres parce que c'est la seule base dont on est à peu près sûr qu'elle est toujours là (mais ça pourrait foirer).

#### #5 - 20 décembre 2018 10:12 - Benjamin Dauvergne

*- Fichier 0001-add-compatibility-layer-for-support-of-Django-native.patch ajouté*

sans le raise en trop.

#### #6 - 20 décembre 2018 10:48 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Hmm ? Je ne comprends pas, tu veux qu'il retourne quoi ?

Ok, je me suis emmêlé les pinceaux.

> C'est pour cela que j'ai mis kwargs, pour ignorer ça, mais je peux mettre un assert not kwargs pour être sûr que ça n'arrive jamais.

Et donc on ne reprend pas le bout de code :

```
        if private_only:
            cls._meta.add_field(self, private=True)
        else:
            cls._meta.add_field(self)
```

de django.db.models.fields.Field.contribute_to_class, qui justifie la présence d'un tel paramètre private_only ?
Pourquoi est-ce qu'on ne pourrait pas avoir de champ JSON privé pour un modèle ?

> Pareil on va dire que ça ne doit pas arriver (définition d'un nom explicite de colonne impliquant la création d'un attribut déferré).

Ok d'ac.

> Parce que sinon ça ne marche pas, voir le code de serialize et deep_deconstruct dans django.db.migrations.autodetector et autour; la méthode deconstruct des Field renvoie un quadruplet mais tous les autres objets des triplets; comme on hérite pas de Field le moteur de migration prend le champ pour un autre type d'objet.

Ok, vu pour la méthode deep_deconstruct, mais pas pour serialize (pas trouvé dans le code db.migrations.autodetector de dj1.11 ni dj1.8).

> Contourner le bug d'incompatibilité entre django.contrib.postgres.fiels.JSONField et jsonfield.JSONField, celui que tu as pointé.

Ok, pardon, relecture trop rapide de ma part.

> Bug tordu, le nouveau champ fait un accès à la base à l'initialisation, et donc en dehors des tests, (pour connaître la version de postgres, mais dès que tu accèdes à django.db.connection même pour lire vendor qui n'est qu'une info coté Django, ça ouvre une connection), comme les settings des tests ne définissait pas NAME ça ne passait pas; l'autre possibilité ce serait de ne pas faire cette accès si DATABASES n'est pas initialisé correctement (il l'est ensuite dans les tests). Et j'ai mis postgres parce que c'est la seule base dont on est à peu près sûr qu'elle est toujours là (mais ça pourrait foirer).

Ok, est-ce qu'on peut mettre en substance ça en commentaire dans ce fichier ?

Je teste le patch maintenant.

#### #7 - 20 décembre 2018 11:31 - Paul Marillonnet

*- Statut changé de Nouveau à En cours*

Ok pour moi.

#### #8 - 20 décembre 2018 11:31 - Paul Marillonnet

*- Tracker changé de Support à Development*

*- Statut changé de En cours à Solution validée*

#### #9 - 20 décembre 2018 12:00 - Paul Marillonnet

(Et donc à toi de voir si on prend en compte le support de champs JSONField en attribut privé de la classe modèle parente.)

#### #10 - 20 décembre 2018 12:02 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Et donc on ne reprend pas le bout de code :
> [...]

Non puisqu'on s'assurerait via assert not kwargs que ça ne peut pas arriver (ou bien ça plante, vu le moment d'initialisation très tôt des modèles, on le ratera pas).

> de django.db.models.fields.Field.contribute_to_class, qui justifie la présence d'un tel paramètre private_only ?
> Pourquoi est-ce qu'on ne pourrait pas avoir de champ JSON privé pour un modèle ?

Je ne répondrai que si tu peux me dire à quoi servent les champs privés/virtuels dans Django.

> > Pareil on va dire que ça ne doit pas arriver (définition d'un nom explicite de colonne impliquant la création d'un attribut déferré).
>
> Ok d'ac.
>
> > Parce que sinon ça ne marche pas, voir le code de serialize et deep_deconstruct dans django.db.migrations.autodetector et autour; la méthode deconstruct des Field renvoie un quadruplet mais tous les autres objets des triplets; comme on hérite pas de Field le moteur de migration prend le champ pour un autre type d'objet.
>
> Ok, vu pour la méthode deep_deconstruct, mais pas pour serialize (pas trouvé dans le code db.migrations.autodetector de dj1.11 ni dj1.8).

Oui c'est ailleurs, dans django.db.migrations.serializer.

> Bug tordu, le nouveau champ fait un accès à la base à l'initialisation, et donc en dehors des tests, (pour connaître la version de postgres, mais dès que tu accèdes à django.db.connection même pour lire vendor qui n'est qu'une info coté Django, ça ouvre une connection), comme les settings des tests ne définissait pas NAME ça ne passait pas; l'autre possibilité ce serait de ne pas faire cette accès si DATABASES n'est pas initialisé correctement (il l'est ensuite dans les tests). Et j'ai mis postgres parce que c'est la seule base dont on est à peu près sûr qu'elle est toujours là (mais ça pourrait foirer).
>
> Ok, est-ce qu'on peut mettre en substance ça en commentaire dans ce fichier ?

Oui tu as raison je vais faire ça, à défaut d'être compréhensible ce sera là.

> Je teste le patch maintenant.

#### #11 - 20 décembre 2018 12:12 - Benjamin Dauvergne

*- Fichier 0001-add-compatibility-layer-for-support-of-Django-native.patch ajouté*

*- Statut changé de Solution validée à Solution proposée*

J'ai ajouté les assert en question, j'ai aussi tenté le coup de ne pas checker la version de la base si pas de NAME, j'aurai pu faire un try/except aussi. J'ai poussé en va voir si ça build bien.

#### #12 - 17 janvier 2019 10:28 - Paul Marillonnet

*- Statut changé de Solution proposée à Solution validée*

Je pensais que j'avais validé et que c'était poussé, pardon.

#### #13 - 18 janvier 2019 19:31 - Benjamin Dauvergne

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

*- % réalisé changé de 0 à 100*

Appliqué par commit authentic2|d730dba525215d615fa2b974d44183d16909f4d2.

#### #14 - 19 janvier 2019 08:47 - Frédéric Péters

Il me semble que ça a cassé le job jenkins d'authentic2-auth-fc, https://jenkins2.entrouvert.org/job/authentic2-auth-fc/119/

#### #15 - 19 janvier 2019 10:36 - Benjamin Dauvergne

*- Fichier 0001-compat-handler-case-of-Django-1.11-without-psycopg2-.patch ajouté*

*- Statut changé de Résolu (à déployer) à Solution proposée*

Effectivement il faut vérifier si la contrib postgres est importable, dans le cas où on fait du sqlite en Django 1.11.

#### #16 - 19 janvier 2019 10:37 - Benjamin Dauvergne

*- Fichier 0001-compat-handle-case-of-Django-1.11-without-psycopg2-f.patch ajouté*

correction au sujet du commit.

#### #17 - 19 janvier 2019 10:41 - Frédéric Péters

*- Statut changé de Solution proposée à Solution validée*

Ack.

#### #18 - 19 janvier 2019 12:10 - Benjamin Dauvergne

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

Appliqué par commit authentic2|000f683601d4b996c2d3dc736c9a4683905dc4ba.

#### #19 - 06 février 2019 12:16 - Frédéric Péters

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

### Fichiers

| | | | |
|---|---|---|---|
| 0001-add-compatibility-layer-for-support-of-Django-native.patch | 9,54 ko | 19 décembre 2018 | Benjamin Dauvergne |
| 0001-add-compatibility-layer-for-support-of-Django-native.patch | 9,52 ko | 20 décembre 2018 | Benjamin Dauvergne |
| 0001-add-compatibility-layer-for-support-of-Django-native.patch | 9,34 ko | 20 décembre 2018 | Benjamin Dauvergne |
| 0001-compat-handler-case-of-Django-1.11-without-psycopg2-.patch | 1,09 ko | 19 janvier 2019 | Benjamin Dauvergne |
| 0001-compat-handle-case-of-Django-1.11-without-psycopg2-f.patch | 1,09 ko | 19 janvier 2019 | Benjamin Dauvergne |