Authentic 2 - Development #22376

API users, avoir la possibilité d'un get_or_create

08 Mar 2018 11:47 AM - Frédéric Péters

Status:	Solution déployée	**Start date:**	08 Mar 2018
Priority:	Normal	**Due date:**	
Assignee:	Benjamin Dauvergne	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
Patch proposed:	Yes	**Planning:**	No

Description

Un formulaire côté wcs qui demande quatre adresses email mais la même peut être réutilisée (parce que c'est genre "adresse du responsable du service A", "... du service B", etc. et que la même personne peut être en charge de différents services); un workflow qui doit créer pour chacune de ces adresses un compte, on peut construire ça avec un appel vérifiant l'existance et permettant de récupérer l'uuid existant si présent puis un autre appel de création si ce n'était pas le cas, mais ce serait plus simple d'avoir un appel type "get or create".

Associated revisions

Revision d03f4fc8 - 21 May 2019 03:59 PM - Benjamin Dauvergne

api: accept get/update_or_create parameter to user and role creation endpoint (fixes #22376)

History

#1 - 08 Mar 2018 02:52 PM - Benjamin Dauvergne

Je verrai plutôt des préconditions, genre :

```
POST /api/users/

{
   ...
   'email': 'whatever@whatever.org',
   'preconditions': {
      'not_exists': [
         {
           'ou': 'agents',
           'email': 'whatever@whatever.org',
         }
       ]
    }
}
```

Si ça ne matche pas je retourne un 412 Precondition failed avec le contenu JSON explicatif.

#2 - 08 Mar 2018 03:07 PM - Frédéric Péters

> avec le contenu JSON explicatif

C'est-à-dire ?

Le truc important avec le get_or_create c'est qu'on récupère ainsi les bonnes infos de l'utilisateur d'une unique manière, qu'on peut ensuite utiliser le long du workflow de manière unique et simple, pour par exemple ensuite appeler un autre webservice en passant identifiant_de_l_appel_create_response_uuid, que l'utilisateur ait été créé ou pas.

#3 - 12 Mar 2018 11:06 AM - Benjamin Dauvergne

Ok bon le gros souci c'est que faire un get_or_create sur des champs sans index d'unicité c'est juste impossible avec Django, le seul moyen c'est de faire comme dans w.c.s.:

- passer en autocommit
- boucler sur
  1. chercher via les champs uniques, si trouvé on retourne
  2. sinon créer un utilisateur, obtenir un id x
  3. faire un select, vérifier si x est l'id le plus petit si plusieurs objets trouvés, sinon supprimer x et recommencer en 1

À voir comment désactiver les transaction et les remettre à la pince à épiler où c'est nécessaire.

#4 - 04 Apr 2018 09:45 AM - Paul Marillonnet

Benjamin Dauvergne a écrit :

> - boucler sur
>   1. chercher via les champs uniques, si trouvé on retourne

Pourquoi via les champs uniques, alors que le get_or_create se fait potentiellement via des champs non uniques ?

> 2. sinon créer un utilisateur, obtenir un id x

Pourquoi commencer par créer l'utilisateur si on est en mode autocommit ?
C'est par souci de cohérence de la base ?
Qu'est-ce que ça change de faire le select et la vérification **après** la création de l'utilisateur ?

> 3. faire un select, vérifier si x est l'id le plus petit si plusieurs objets trouvés, sinon supprimer x et recommencer en 1

Ok

#5 - 04 Sep 2018 02:46 PM - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Benjamin Dauvergne a écrit :
>
> - boucler sur
>   1. chercher via les champs uniques, si trouvé on retourne
>
> Pourquoi via les champs uniques, alors que le get_or_create se fait potentiellement via des champs non uniques ?

get_or_create() ne garantit pas l'unicité sur des champs non unique (il faut au moins couvrir un index d'unicité dans les critères, soit mono champ soit multi champ mais il faut le couvrir, i.e. que toutes ses colonnes soient citées).

>   2. sinon créer un utilisateur, obtenir un id x
>
> Pourquoi commencer par créer l'utilisateur si on est en mode autocommit ?
> C'est par souci de cohérence de la base ?
> Qu'est-ce que ça change de faire le select et la vérification **après** la création de l'utilisateur ?

C'est pour simuler du verrouillage optimise sans verrou, en vrai c'est limite sans transactions c'est quasiment impossible d'assurer l'unicité sans index sauf à avoir une limite haute sur le temps d'exécution des différents clients.

>   3. faire un select, vérifier si x est l'id le plus petit si plusieurs objets trouvés, sinon supprimer x et recommencer en 1
>
> Ok

En fait en autocommit il vaudrait mieux boucler sur quelque chose comme cela (exécution concurrente possible):

Temps	Client 1	Client 2
1	COUNT x=1 -> 0	
2		COUNT x=1 -> 0
3	INSERT x=1 -> id=a	
4		INSERT x=1 -> id=b
5	COUNT x=1 -> 2	
6		COUNT x=1 -> 2
7	DELETE id=a	
8	wait random() ms; go to 1	DELETE id=b
9		wait random() ms; go to 1

On fait un select pour voir si ce qu'on veut insérer est là, sinon on insère, on revérifie, sin on plus d'une ligne, on supprime celle qu'on vient d'insérer et on recommence en attendant un délai aléatoire (c'est un peu comme Ethernet, https://fr.wikipedia.org/wiki/Carrier_Sense_Multiple_Access_with_Collision_Detection).

C'est plus sûr que mon idée de garder l'id le plus petit (dans le cas où c'est le client avec l'id le plus grand qui fait la vérification en premier, i.e. si 6 se déroule avant 5 et que b > a alors ça ne marche pas, il vaut mieux dans tous le cas que celui qui détecte le doublon ré-essaye).

Ça marche parce qu'on est sûr qu'en autocommit 3 ou 4 s'exécute en premier et entièrement, donc en cas d'exécution entrecoupées, un des deux

clients verra forcément un doublon.

On peut avoir un doublon en cas de crash entre le INSERT et la deuxième vérification, mais c'est plus improbable.

Il ne faut surtout pas exécuter tout cela dans une transaction parce qu'avec le niveau d'isolation par défaut de postgres (READ COMITTED) et bien les deux transactions peuvent faire tout cela sans se voir (READ COMITTED, on ne voit que les écritures "comittés" pendant la transaction).

#6 - 05 Sep 2018 09:48 AM - Paul Marillonnet

- Assignee set to Paul Marillonnet

#7 - 05 Sep 2018 08:14 PM - Paul Marillonnet

Merci Benjamin pour ces explications. Je vais partir là dessus.

Et oui en effet, il y a du code similaire dans wcs, je viens de découvrir cela.
Par exemple ici :
https://git.entrouvert.org/wcs.git/tree/wcs/sql.py#n831

Edit: ici l'exemple choisi n'est peut-être pas le meilleur, parce que le passage en mode autocommit se fait dans la fonction appelante.
Mais l'idée est là je crois.

#8 - 20 Sep 2018 04:11 PM - Paul Marillonnet

J'ai du mal à m'imaginer quelle serait l'interface viable et implémentable à l'aide du framework REST Django.

Benjamin, est-ce que ta première remarque dans ce ticket reste valable ? (introduire un champ 'preconditions' dans le payload JSON -- auquel cas, ce serait plutôt un "create_or_get")

Si oui, est-ce envisageable du point de vue de la cohérence de l'API, de renvoyer dans le payload, en plus du message explicatif de la réponse HTTP 412, le JSON de l'utilisateur déjà existant ?

Ou, alternative, virer ce champ "preconditions", et patcher l'API users pour renvoyer un HTTP 409 Conflict, en réponse au POST si l'utilisateur existe déjà, avec en payload le JSON correspondant à cet utilisateur ?

#9 - 01 Oct 2018 09:44 AM - Paul Marillonnet

Je voudrais bien partir, côté API, sur l'option "renvoyer un HTTP 409 avec un JSON décrivant le compte utilisateur, lorsque l'on tente de le créer alors qu'il existe déjà".
Et donc, pour faire un "get or create" côté appel WS, il faudra faire un POST avec les champs de l'usager que l'on souhaite créer s'il n'existe as déjà (ce serait donc plutôt un "create or get").
On se ramène bien à un seul appel. Est-ce c'est bon ou bien ça pose problème côté w.c.s. ?

#10 - 01 Oct 2018 03:03 PM - Paul Marillonnet

Discussion avec Thomas là-dessus.
Est-ce qu'on pourrait rajouter, dans le dico envoyé à l'API, un champ destiné à authentic2.api_views.BaseUserSerializer.create ?
Par exemple :

```
{
    'email': 'pmar@eo.org',
    'last_name': '...',

    'get_or_create_by': 'email', <---- provoque un User.objects.get_or_create(email='pmar@eo.org', defaults=va
lidated_data)

}
```

#11 - 01 Oct 2018 03:25 PM - Benjamin Dauvergne

Je serai plus pour un ou des paramètres dans la query string, genre ?get-or-create-email=xyz@example.com&get-or-create-ou__slug=zob, juste qu'il va falloir parser tout ça et vérifier que c'est juste (que ou__slug ça existe, ou alors faire une whitelist dans un premier temps, username, email et ou__slug devraient suffire).

#12 - 05 Oct 2018 04:04 PM - Paul Marillonnet

- Status changed from Nouveau to Solution proposée

- File 0001-add-get-or-create-users-api-call-22376.patch added

- Patch proposed changed from No to Yes

Benjamin Dauvergne a écrit :

> Je serai plus pour un ou des paramètres dans la query string, genre ?get-or-create-email=xyz@example.com&get-or-create-ou__slug=zob, juste qu'il va falloir parser tout ça et vérifier que c'est juste (que ou__slug ça existe, ou alors faire une whitelist dans un premier temps, username, email et ou__slug devraient suffire).

Voilà ma compréhension du truc. Est-ce que c'est bien ça que tu voulais dire ?

#13 - 15 Mar 2019 03:19 AM - Benjamin Dauvergne

- File 0001-api-accept-get_or_create-parameter-to-user-creation-.patch added

Autre approche plus simple je pense, adaptable à tous les modèles.

Plutôt que de rendre le truc implicite dans Serializer.create() je pourrai
aussi adapter la vue pour qu'elle fasse le choix entre create/get_or_create()
et gère le retour "created" pour décider d'une réponse 200 ou 201.

#14 - 20 Mar 2019 10:45 AM - Paul Marillonnet

Peux-tu m'expliquer la nécessité de discerner entre old_value et new_value, en fonction du thread d'exécution, dans la monkeypatch_method stp ? Je ne comprends pas en quoi le fait d'avoir basculé sur un autre thread nécessite de retrouver l'ancienne valeur de l'attribut "monkeypatché" ? Pour

moi si l'on veut éviter les race conditions il faut surtout laisser cette ancienne valeur en paix
(J'ai l'impression que le code présume que si le thread actuel n'est plus celui identifié par current_thread_id, alors c'est ce thread actuel qui effectue le premier get_or_create. Je crois que je loupe quelque chose, ça n'est pas encore clair pour moi.)

À part ça oui je pense que ce serait mieux de pouvoir renvoyer 201 ou 200 en fonction de la création ou non de l'objet.

#15 - 20 Mar 2019 11:12 AM - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Peux-tu m'expliquer la nécessité de discerner entre old_value et new_value, en fonction du thread d'exécution, dans la monkeypatch_method stp ?

Sinon on modifie la méthode dans les autres threads.

> Je ne comprends pas en quoi le fait d'avoir basculé sur un autre thread nécessite de retrouver l'ancienne valeur de l'attribut "monkeypatché" ? Pour moi si l'on veut éviter les race conditions il faut surtout laisser cette ancienne valeur en paix

Parce que le comportement est dépendant du thread d'exécution si on traite une requête par thread.

> (J'ai l'impression que le code présume que si le thread actuel n'est plus celui identifié par current_thread_id, alors c'est ce thread actuel qui effectue le premier get_or_create. Je crois que je loupe quelque chose, ça n'est pas encore clair pour moi.)

Il n'y a ni premier ni rien, les méthodes sont des objets globaux, en présence de thread on ne peut pas les modifier comme cela sans être certain de ce qu'on veut; en l'étant ce n'est pas non plus ré-entrant (mais heureusement on n'a pas d'appel à create() depuis create()).

> À part ça oui je pense que ce serait mieux de pouvoir renvoyer 201 ou 200 en fonction de la création ou non de l'objet.

Yep.

#16 - 20 Mar 2019 11:51 AM - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Sinon on modifie la méthode dans les autres threads.

Ah bein oui, du coup c'est plus clair. Merci.

Rien à voir, et c'est du pinaillage, mais étant donné le caractère très générique de la chose, les appels au manager du modèle dans le mixin gagneraient à de pas présumer de l'existence de model.objects.
(Cf https://docs.djangoproject.com/fr/1.11/topics/db/managers/#default-managers : *"Si par exemple vous écrivez du code qui doit gérer un modèle inconnu dans une application tierce qui implémente une vue générique, utilisez ce gestionnaire (ou _base_manager) plutôt que de supposer que le modèle possède un gestionnaire objects."*)

#17 - 01 Apr 2019 11:49 AM - Benjamin Dauvergne

- Assignee changed from Paul Marillonnet to Benjamin Dauvergne

#18 - 01 Apr 2019 11:50 AM - Benjamin Dauvergne

- Status changed from Solution proposée to En cours

Je vais virer les bidouilles affreuses.

#21 - 21 May 2019 01:40 PM - Benjamin Dauvergne

- Status changed from En cours to Solution proposée

- File 0001-api-accept-get-update_or_create-parameter-to-user-an.patch added

Voilà je suis obligé de recopier du code de django-rest-framework mais tout est plus clair, au passage on gagne aussi un update_or_create.

À noter que pour les utilisateurs en utilisant l'email on n'aura pas la garantie de ne pas avoir de doublon car on a pas d'index d'unicité sur le champ email.

#22 - 21 May 2019 03:48 PM - Paul Marillonnet

Quelques modifs qui me paraissent utiles voire nécessaires (surtout les dernières, en fin de diff) :

```
diff --git a/src/authentic2/api_mixins.py b/src/authentic2/api_mixins.py
index fe51f4e4..2c1d8754 100644
--- a/src/authentic2/api_mixins.py
+++ b/src/authentic2/api_mixins.py
@@ -46,7 +46,7 @@ class GetOrCreateModelSerializer(object):
             else:
                 defaults[key] = value
         with transaction.atomic():
-            instance, created = self.Meta.model.objects.get_or_create(**kwargs)
+            instance, created = self.Meta.model._default_manager.get_or_create(**kwargs)
             if many_to_many and created:
                 self.update(instance, many_to_many)
         return instance
```

```
@@ -76,9 +76,11 @@ class GetOrCreateModelSerializer(object):
             else:
                 defaults[key] = value
         with transaction.atomic():
-            instance, created = self.Meta.model.objects.get_or_create(**kwargs)
-            if many_to_many or not created:
+            instance, created = self.Meta.model._default_manager.get_or_create(**kwargs)
+            if not created:
                 self.update(instance, validated_data)
+            if many_to_many:
+                self.update(instance, many_to_many)
         return instance

     def create(self, validated_data):
```

De façon optionnelle, peut-être aussi un test avec un get_or_create ou un update_or_create sur plusieurs clés ?
Peut-être aussi une méthode "privée" (pas au sens OOP) pour factoriser tout le code identique des deux méthodes {get,update}_or_create introduites (en gros tout jusqu'au with transaction.atomic() ?

#23 - 21 May 2019 03:54 PM - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Quelques modifs qui me paraissent utiles voire nécessaires (surtout les dernières, en fin de diff) :

Ok pour le _default_manager, merci, pour le reste je vais faire à ma sauce.

> De façon optionnelle, peut-être aussi un test avec un get_or_create ou un update_or_create sur plusieurs clés ?

Ok.

> Peut-être aussi une méthode "privée" (pas au sens OOP) pour factoriser tout le code identique des deux méthodes {get,update}_or_create introduites (en gros tout jusqu'au with transaction.atomic() ?

Je voulais au début mais le code est plus clair comme ça et ressemble à celui de create() dans rest_framework.

#24 - 21 May 2019 03:59 PM - Benjamin Dauvergne

- File 0001-api-accept-get-update_or_create-parameter-to-user-an.patch added

Done.

#25 - 21 May 2019 04:08 PM - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Je voulais au début mais le code est plus clair comme ça et ressemble à celui de create() dans rest_framework.

Ok.

Et il me semble avoir ramassé un bug dans la suite du diff :

```
-             if many_to_many or not created:
+             if not created:
                  self.update(instance, validated_data)
+             if many_to_many:
+                 self.update(instance, many_to_many)
          return instance
```

Je me plante ?

Edit:
Ce qui avec ton nouveau patch serait plutôt :

```
-             if many_to_many or not created:
-                 self.update(instance, validated_data)
+             if not created:
+                 self.update(instance, get_or_create_data)
+             if many_to_many:
+                 self.update(instance, many_to_many)
```

#26 - 21 May 2019 04:40 PM - Benjamin Dauvergne

Ce n'est pas nécessaire et on économise un double .save() car .update() refait la séparation many_to_many/autres champs.

#27 - 21 May 2019 05:03 PM - Paul Marillonnet

- Status changed from Solution proposée to Solution validée

Benjamin Dauvergne a écrit :

> Ce n'est pas nécessaire et on économise un double .save() car .update() refait la séparation many_to_many/autres champs.

Ok, moins lisible àmha, mais ok.

#28 - 21 May 2019 05:20 PM - Benjamin Dauvergne

- Status changed from Solution validée to Résolu (à déployer)

```
commit d03f4fc8d3da9c061f52f01580ecd5f733c977c9
Author: Benjamin Dauvergne <bdauvergne@entrouvert.com>
Date:   Fri Mar 15 03:17:04 2019 +0100

    api: accept get/update_or_create parameter to user and role creation endpoint (fixes #22376)
```

#29 - 21 May 2019 05:20 PM - Benjamin Dauvergne

- % Done changed from 0 to 100

Appliqué par commit authentic2|d03f4fc8d3da9c061f52f01580ecd5f733c977c9.

#30 - 24 May 2019 06:15 PM - Frédéric Péters

- Status changed from Résolu (à déployer) to Solution déployée

Files

0001-add-get-or-create-users-api-call-22376.patch	6.12 KB	05 Oct 2018	Paul Marillonnet
0001-api-accept-get_or_create-parameter-to-user-creation-.patch	5.52 KB	15 Mar 2019	Benjamin Dauvergne
0001-api-accept-get-update_or_create-parameter-to-user-an.patch	9.49 KB	21 May 2019	Benjamin Dauvergne
0001-api-accept-get-update_or_create-parameter-to-user-an.patch	10.6 KB	21 May 2019	Benjamin Dauvergne