Authentic 2 - Development #20706

API de création de rôle

14 décembre 2017 14:28 - Frédéric Péters

Statut:	Fermé	**Début:**	14 décembre 2017
Priorité:	Normal	**Echéance:**	
Assigné à:	Paul Marillonnet	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Je voudrais depuis un workflow de w.c.s. pouvoir créer de nouveaux rôles (après un formulaire "déclaration d'une association" créer automatiquement un rôle où pourraient être rangés les membres de celle-ci); il ne semble pas y avoir d'API pour ça.

Demandes liées:

Lié à Authentic 2 - Development #21488: inclure les attributs de rôle dans l'API	**Fermé**	**29 janvier 2018**
Lié à Authentic 2 - Development #22251: API de création de rôle : calcul auto...	**Fermé**	**04 mars 2018**

Révisions associées

Révision eef27f83 - 05 mars 2018 17:42 - Paul Marillonnet

rename role membership API class (pre-#20706)

Révision 85da1be8 - 05 mars 2018 17:46 - Paul Marillonnet

add role-creation API (#20706)

Historique

#1 - 15 décembre 2017 17:39 - Paul Marillonnet

Dans le constructeur de la classe RolesAPI de api_views, je vois une ligne :

```
    self.role = get_object_or_404(Role, uuid=kwargs['role_uuid'])
```

Donc potentiellement très peu de code à changer pour que le rôle soit créé, non ?

#2 - 15 décembre 2017 17:50 - Frédéric Péters

Ça peut vraisemblablement être autour de la classe RolesAPI existante mais c'est un peu de code quand même. (cf sans doute UsersAPI et la partie BaseUserSerializer et la classe ModelViewSet).

#3 - 21 décembre 2017 15:18 - Benjamin Dauvergne

- Assigné à mis à Benjamin Dauvergne

Gogogo.

#4 - 22 décembre 2017 16:01 - Benjamin Dauvergne

- Assigné à changé de Benjamin Dauvergne à Paul Marillonnet

Dans un excès d'enthousiasme je me suis auto-assigné, il fallait bien sur comprendre que je passais la chose à Paul.

#5 - 22 décembre 2017 16:12 - Paul Marillonnet

Ok je prends.

#6 - 22 décembre 2017 16:17 - Paul Marillonnet

Donc ici, pour reprendre ce qui a été dit dans #16523 et #20454, il faut toujours rattacher le rôle nouvellement créé à un service et à une OU, on est bien d'accord ?

#7 - 22 décembre 2017 16:52 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Donc ici, pour reprendre ce qui a été dit dans #16523 et #20454, il faut toujours rattacher le rôle nouvellement créé à un service et à une OU, on est bien d'accord ?

Oui et c'est là ou je me dis que je repartirai bien d'un autre mapping d'URL que l'actuel, genre

/api/ous/<ou_id_or_ou_slug>/roles/ comme ça on rend le fait de choisir une OU obligatoire, progressivement on recollerai UsersAPI sous ces nouvelles URLs d'OU, pour l'instant les OUs n'était pas bien représenté dans les APIs.

C'est vraiment discutable, on peut aussi se contenter d'un POST sur /api/roles/ avec une clé OU obligatoire et voir ce que je dis plus tard. Le rattachement à un service j'ignorerai ça pour l'instant, c'est spécifique au fonctionnement de Publik/Hobo/etc.. essayons de ne pas coupler ça avec les APIs pour l'instant.

#8 - 04 janvier 2018 09:26 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

- Patch proposed changé de Non à Oui

Je suis en train de jouer avec le framework REST de Django, que je connais pas bien.

Comme le montre le patch WIP de 'pseudocode' joint ici, je doute maintenant qu'il faille réutiliser api_views.RolesAPI, qui est en charge de l'assignation des utilisateurs dans les rôles.

J'ai donc renommé cette classe en RoleMembershipsAPI.
Je pense opter pour un format d'URL /api/ous/<ou_id_or_ou_slug>/roles/ comme conseillé par Benj, et donc je voudrais créer une CBV nommée RolesInOuAPI (pour insister sur la nécessité de choisir une OU pour toute manipulation de rôle via l'API).

#9 - 04 janvier 2018 10:17 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

Commencé à écrire du pseudocode pour la CBV, notamment sur la création et suppression de rôle, mais je ne sais pas comment utiliser correctement les serializers. Je lis la doc du DRF là-dessus.

#10 - 04 janvier 2018 10:19 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch supprimé

#11 - 04 janvier 2018 10:20 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

#12 - 08 janvier 2018 15:53 - Benjamin Dauvergne

Tu peux appeler ta CBV RolesAPI, en renommant l'existante comme tu le suggères, pas besoin d'indiquer InOU.

#13 - 09 janvier 2018 15:52 - Paul Marillonnet

Est-ce que j'implémente directement un API CRUD pour les rôles, ou bien la création seule est pertinente ici ?

#14 - 09 janvier 2018 15:57 - Frédéric Péters

De mon côté j'ai vraiment juste besoin de créer. (même si django-rest-framework offre le reste)

#15 - 09 janvier 2018 17:03 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

Est-ce que je reprends les classes offertes par le framework REST, ou bien j'essaie d'avoir le moins de modification dans le code existant ?

Si réponse B, on pourrait peut-être partir sur un patch dont le pseudocode serait comme ci-joint (pas encore testé), et qui donc ne reprendrait pas les mécanismes offerts par le DRF (ModelViewSet, routeurs, serializers, ...) -- ceux-ci paraissant faciliter les choses à condition de vouloir une API CRUD complète, non ?

#16 - 09 janvier 2018 22:08 - Benjamin Dauvergne

- ça m'irait de séparer dans un premier commit le renommage de la classe existante
- je suis pas fan des exceptions qui ne servent qu'une fois surtout que tu peux directement faire un return Response() à la place
- DRF fournit la classe Serializer et ModelSerializer pas besoin de _api_fetched_fields
- je mettrai plutôt ou-id-or-slug dans l'URL (kwargs)

- je ne vois pas le besoin A2_ROLES_REQUIRED_FIELDS

À noter qu'il y a une difficulté, dans le cadre de Publik le modèle des Rôles est étendu de deux attributs stockés dans des modèles RoleAttribute voir hobo/agent/authentic/role_forms.py, nommé details (str), emails(list of str) et emails_to_members(boolean) stockés en JSON, à Fred de dire si c'est important pour lui de pouvoir définir ces attributs à ce stade.

#17 - 09 janvier 2018 22:31 - Frédéric Péters

> À noter qu'il y a une difficulté, dans le cadre de Publik le modèle des Rôles est étendu de deux attributs stockés dans des modèles RoleAttribute voir hobo/agent/authentic/role_forms.py, nommé details (str), emails(list of str) et emails_to_members(boolean) stockés en JSON, à Fred de dire si c'est important pour lui de pouvoir définir ces attributs à ce stade.

Pas particulièrement, non.

#18 - 12 janvier 2018 10:59 - Paul Marillonnet

- Fichier 0001-rename-role-membership-API-class-pre-20706.patch ajouté

Benjamin Dauvergne a écrit :

- ça m'irait de séparer dans un premier commit le renommage de la classe existante.

#19 - 12 janvier 2018 12:12 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

Benjamin Dauvergne a écrit :

- je suis pas fan des exceptions qui ne servent qu'une fois surtout que tu peux directement faire un return Response() à la place
- je mettrai plutôt ou-id-or-slug dans l'URL (kwargs)
- je ne vois pas le besoin A2_ROLES_REQUIRED_FIELDS

Remarques prises en compte dans le patch WIP.
Pour faire tourner le code je voudrais commencer par écrire les tests (comme le premier test fourni dans le patch).

Je vais regarder comment sont 'mockées' les OUs dans les tests.

- DRF fournit la classe Serializer et ModelSerializer pas besoin de _api_fetched_fields

OK je regarde ça, et je retire ce _api_fetched_fields dans le prochain patch.

#20 - 12 janvier 2018 15:49 - Paul Marillonnet

Pour l'instant tout ce que j'arrive à avoir de mes tests, c'est une erreur de jeton CSRF manquant.
Je ne vois nulle part une config CSRF dans les requête émises à l'aide de django_webtest.DjangoTestApp

J'ai beau décorer ma méthode RolesAPI.post() comme ci-dessous :

```
+    @method_decorator(csrf_exempt)
     def post(self, request, *args, **kwargs):
```

pour l'instant rien n'y fait.
Je continue de creuser l'affaire.

#21 - 12 janvier 2018 16:40 - Benjamin Dauvergne

Probablement que tu fais des post et pas des post JSON et donc tu passes par la vue HTML et non pas par la vue web-service.

#22 - 12 janvier 2018 16:53 - Paul Marillonnet

Je me mange les mêmes erreur avec post_json, bizarre bizarre.
Mais oui je comprends ta remarque. Je vais vérifier, si ça se trouve mon code pour RolesAPI.post() n'est même pas exécuté...
Je regarde ça.

(Au pire pour en avoir le cœur net, django_webtest.WebTestMixin propose un champ csrf_checks)

#23 - 15 janvier 2018 10:07 - Paul Marillonnet

Le champ csrf_checks du WebTestMixin n'y change rien, étrange.

Je vais regarder d'où est remontée l'erreur CSRF et partir de là.

Edit : l'erreur vient de django.middleware.csrf.CsrfViewMiddle.process_view() et pourtant impossible de prendre la main sur cette partie du code à l'aide du débogueur Python, comme si elle n'était pas exécutée.
Je continue de chercher.

Edit 2 : Je me demande si les modifs dans mon code ne provoquent pas une erreur à l'initialisation des fixtures, puisque je n'arrive pas à prendre la main sur la levée d'erreur dans les code de test. (La trace de la pile d'exécution est vraiment pas explicite)
Je vais creuser aussi cette piste.

#24 - 15 janvier 2018 14:20 - Paul Marillonnet

Bon, c'est APIView.initial() du DRF qui semble vérifier la présence du CSRF, parce qu'après un modif pas très propre du genre :

```
@@ -582,6 +584,9 @@ class RolesAPI(ExceptionHandlerMixin, APIView):

+    def initial(self, request, *args, **kwargs):
+        pass
+
     def post(self, request, *args, **kwargs):
         logger = logging.getLogger(__name__)
```

l'erreur CSRF n'est plus levée.
Je vais reprendre la méthode initial() de l'ancienne classe RolesAPI (renommée en RoleMembershipsAPI).

Je me penche maintenant sur l'implémentation du sérialisateur (en héritant le ModelSerializer du DRF).

#25 - 15 janvier 2018 15:01 - Benjamin Dauvergne

Sincèrement le problème est certainement entre la chaise et le clavier, si tu regardes les autres vues et leur test tu verras que rien n'est fait pour un quelconque check CSRF qui n'existe pas sur les web-services, il y a quelque chose que tu fais mal dans tes tests, forcément.

#26 - 15 janvier 2018 15:03 - Benjamin Dauvergne

À mon avis c'est parce que tu utlises logged_app qui n'est utilisé par aucune autre test à part le test de la vue JSONP des informations utilisateur, les autres tests font du HTTP Basic.

#27 - 15 janvier 2018 16:44 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Sincèrement le problème est certainement entre la chaise et le clavier,

Aucun doute là dessus. Je note mes réflexions ici au fur et à mesure (dans une démarche d'auto-diagnostic) pour comprendre ce qui ne va pas dans ma manière d'aborder le problème.

> si tu regardes les autres vues et leur test tu verras que rien n'est fait pour un quelconque check CSRF qui n'existe pas sur les web-services, il y a quelque chose que tu fais mal dans tes tests, forcément.

Oui c'est justement ça qui me laisser perplexe.

> À mon avis c'est parce que tu utlises logged_app qui n'est utilisé par aucune autre test à part le test de la vue JSONP des informations utilisateur, les autres tests font du HTTP Basic.

Oui, ma faute : je ne comprends pas encore comment sont implémentés l'authentification et le contrôle d'accès aux APIs A2.
Je vais regarder ça, merci pour le tuyau.

#28 - 15 janvier 2018 17:31 - Paul Marillonnet

Je laisse tomber la classe APIView fournie par le DRF, que je croyais à tort plus pertinente pour la fonctionnalité à implémenter.

J'ai l'impression qu'avec ModelViewSet il n'y a pas ou peu besoin de redéfinir les méthodes de cette classe (create, update, destroy, ...) à condition d'implémenter correctement les vérifications effectuées par le sérialisateur.

#29 - 15 janvier 2018 19:09 - Benjamin Dauvergne

Tout à fait, si tu peux utiliser les classes de haut niveau de DRF ne t'en prive pas.

#30 - 15 janvier 2018 20:25 - Paul Marillonnet

Pour les sérialisateurs de classe d'objets liés à des OU, je vois des SlugRelatedField (et même un début BaseOrganizationalUnitSerializer dans le code).

Je ne comprends pas pourquoi on ne peut pas se contenter d'un CharField pour stocker le slug de l'OU, à condition de vérifier l'existence de l'OU lors de la sérialisation, et d'effectuer le lien objet <-> OU lors de la désérialisation.

#31 - 15 janvier 2018 20:46 - Paul Marillonnet

Je ne trouve pas non plus la bonne façon d'indiquer au sérialisateur que l'identifiant de l'OU ne se trouve pas dans le payload JSON de la requête, mais doit au contraire être allé chercher dans le contexte de la requête (en tant que paramètre dans l'URI). Je suis en train de bidouiller un truc dans le constructeur du sérialisateur, mais je doute que ce soit la bonne approche.

Edit : je vais essayer le paramètre source=<nom de la fonction ou méthode> lors de la construction d'un serializers.CharField.

#32 - 16 janvier 2018 10:37 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Pour les sérialisateurs de classe d'objets liés à des OU, je vois des SlugRelatedField (et même un début BaseOrganizationalUnitSerializer dans le code).
>
> Je ne comprends pas pourquoi on ne peut pas se contenter d'un CharField pour stocker le slug de l'OU, à condition de vérifier l'existence de l'OU lors de la sérialisation, et d'effectuer le lien objet <-> OU lors de la désérialisation.

Le SlugRelatedField évite justement de gérer tout ça tu as deux types de champs pour les ForeignKey, ModelField qui va utiliser la clé primaire comme sérialisation du lien ou SlugRelatedField qui va utiliser en champ slug (qui bien sûr devra être unique pour pouvoir jouer le rôle d'une clé primaire). Ça me va de continuer que l'interface entre l'extérieur et les OU c'est leur slug (mais faudrait certainement le rendre visible et éditable à un moment via /manage/ actuellement ce n'est possible que via /admin/).

#33 - 16 janvier 2018 10:42 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Je ne trouve pas non plus la bonne façon d'indiquer au sérialisateur que l'identifiant de l'OU ne se trouve pas dans le payload JSON de la requête, mais doit au contraire être allé chercher dans le contexte de la requête (en tant que paramètre dans l'URI). Je suis en train de bidouiller un truc dans le constructeur du sérialisateur, mais je doute que ce soit la bonne approche.
>
> Edit : je vais essayer le paramètre source=<nom de la fonction ou méthode> lors de la construction d'un serializers.CharField.

T'as 3 appels à gérer:

- GET /ous/<ou_slug>/roles/ : liste les rôles de l'OU, suffit de jouer sur le get_queryset() de la vue je pense
- POST /oust/<ou_slug>/roles/ : suffit de passer l'OU au constructeur du sérialiseur et de s'en servir dans Serializer.create() voir BaseUserSerializer.create()
- GET /ous/<ou_slug>/roles/<whatever_id>/ : récupérer la description d'un rôle, et là en fait si tu as modifier get_queryset() pour le premier ça va déjà marcher quelquesoit ce que tu utilisers pour whatever_id (le slug du rôle qui est unique uniquement pour une OU, ou l'uuid ou la clé primaire qui sont globalement uniques)

#34 - 16 janvier 2018 12:50 - Paul Marillonnet

Ok d'ac.

Je ne m'en sors pas avec la doc du DRF, j'ai commencé à lire le code de ModelViewSet, de ModelSerializer et des mixins associés.

Edit : c'est beaucoup mieux en lisant le code. Je pose un patch cet ApM.

#35 - 16 janvier 2018 14:01 - Benjamin Dauvergne

Oui sans lire le code ce n'est pas utilisable (mais pour moi c'est pareil pour Django souvent).

#36 - 16 janvier 2018 16:41 - Serghei Mihai

Peut-être que http://www.cdrf.co/ pourrait être utile.

#37 - 16 janvier 2018 18:07 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

Serghei Mihai a écrit :

> Peut-être que http://www.cdrf.co/ pourrait être utile.

Bien vu, merci.

Après m'être douloureusement noyé dans la doc, et surtout après avoir lu le code, je comprends un peu mieux le framework.

Un premier patch WIP avec un scénario de POST minimal sur l'API, que je vais venir compléter ensuite.

Sur ma todolist :
- sérialisation correcte des champs complexes (service, ou, admin_scope_ct)
- récupération de la liste des rôles
- récupération de la description d'un rôle

#38 - 16 janvier 2018 18:11 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

Erreur d'indentation dans le constructeur du sérialisateur, je corrige ici.

#39 - 18 janvier 2018 16:21 - Paul Marillonnet

Frustrant.
La création du rôle se passe bien, mais après avoir appelé la méthode Role.save(), le lien FK à l'OU est perdu, je ne comprends pas pourquoi.

#40 - 18 janvier 2018 18:41 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-api-20706.patch ajouté

Je n'arrive pas à diagnostiquer le problème ce soir.

Mon patch WIP, toujours ce probléme de déréférencement de la ForeignKey OU lors de la sauvegarde du rôle en base. Je m'arracherai les cheveux là-dessus demain.

Je passe à l'implémentation du support des méthodes GET pour la récupération des listes de rôles et de la description d'un rôle une fois fourni son slug.

#41 - 18 janvier 2018 21:35 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-API-20706.patch ajouté

Paul Marillonnet a écrit :

> Je passe à l'implémentation du support des méthodes GET pour la récupération des listes de rôles et de la description d'un rôle une fois fourni son slug.

Début de l'implémentation dans le patch.
Me reste avant tout à corriger ce problème de référence à l'OU non sauvegardé, et à voir comment sérialiser le admin_scope_ct.

#42 - 19 janvier 2018 10:50 - Paul Marillonnet

Paul Marillonnet a écrit :

> toujours ce problème de déréférencement de la ForeignKey OU lors de la sauvegarde du rôle en base. Je m'arracherai les cheveux là-dessus demain.

Deux pistes pour l'instant :
- utiliser le backend postgres pour les tests à la place de SQLite
- http://www.django-rest-framework.org/api-guide/relations/#custom-hyperlinked-fields doc que je n'avais pas vue jusque là.

#43 - 19 janvier 2018 11:31 - Paul Marillonnet

J'ai maintenant l'impression que les champs role.admin_scope_{ct,id} ne doivent pas être configurable via l'API.
Pas très bien compris à quoi correspondant l'ID, mais en tout cas ça me paraît normal que le ContentType reste le même, càd celui décrivant les rôles dans la base A2 (celui dont l'app_label et le name valent respectivement a2_rbac et role). Cette initialisation des champs admin_scope* n'est pas effectuée par le constructeur de Role, et je ne sais pas si l'API doit s'en charger.

Benj, je me plante là-dessus ?

#44 - 19 janvier 2018 13:16 - Benjamin Dauvergne

Popopo qu'est-ce que c'est que ça :)

Bon on va abandonner l'idée de mettre /ou/<slug>/ en racine et on va tout mettre dans /roles/ ça va simplifier ton routage car tu vas pouvoir

simplement enregistrer ta vue dans le routeur principal (router.register).

Tu n'as pas à implémenter les méthode du serializer il te les fournit (create, update, etc..) ni à déclarer les champs du modèle (il fait ça tout seul).

Pas besoin d'initialiser le slug, la classe django_rbac.models.AbstractBase le fait déjà.

Il faut cacher les champs service, admin_scope_*.

Voilà avec tout ça tu devrais pouvoir supprimer 90% du code de ton patch.

#45 - 19 janvier 2018 16:32 - Paul Marillonnet

Ok je reprends là-dessus, ce sera plus simple oui. Merci pour ton aide.

(Les deux pistes que j'ai mentionnées plus haut se sont avéré être fausses, notamment hériter de HyperlinkedRelatedField, qui ne semble pas du tout être la bonne approche).

#46 - 19 janvier 2018 18:41 - Paul Marillonnet

- Fichier 0001-add-role-creation-API-20706.patch ajouté

C'est plus simple comme ça :)

Edit: ajouté un assert len(resp.json['results']) dans le test d'obtention de la liste des rôles, pour que le test échoue si A2 ne renvoie rien.

#47 - 19 janvier 2018 18:56 - Benjamin Dauvergne

Ça ça ne sert à rien:

```
    def get_queryset(self):
        Role = get_role_model()
        return Role.objects.all()
```

Il faut juste déclarer le modèle au niveau de la vue.

Je mettrai uuid en read-only field, on ne souhaite pas que les gens le choisissent.

#48 - 19 janvier 2018 18:57 - Benjamin Dauvergne

Aussi il manque encore toute la partie permission, IsAuthenticated c'est un peu faible :) Regarde ce qui est fait dans les autres vues.

#49 - 19 janvier 2018 19:42 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Il faut juste déclarer le modèle au niveau de la vue.

Je crois que c'est le queryset qu'il faut déclarer au niveau de la vue (il a un `assert self.queryset is not None` dans la méthode get_queryset par défaut).

Je retire la méthode et déclare un attribut queryset à la place.

> Je mettrai uuid en read-only field, on ne souhaite pas que les gens le choisissent.

OK

> Aussi il manque encore toute la partie permission, IsAuthenticated c'est un peu faible :) Regarde ce qui est fait dans les autres vues.

Je vais regarder, merci.

#50 - 19 janvier 2018 19:47 - Paul Marillonnet

- Fichier 0001-add-role-creation-API-20706.patch ajouté

(Je mets ici la dernière version à jour, sans la partie permissions, au cas où ça peut quand même excuser auprès de Frédéric mon retard !)

#51 - 20 janvier 2018 12:15 - Paul Marillonnet

Commencé à implémenter une vérification des permissions dans le ModelSerializer, mais je me dis maintenant il vaut peut-être mieux le faire au plus tôt, dans le ModelViewSet.

#52 - 22 janvier 2018 18:09 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-API-20706.patch ajouté

Voilà, le contrôle d'accès sur la création et la modification des rôles faite dans le sérialisateur.

Je vais regarder si faire la même chose (AC dans le ModelSerializer) est faisable et simple pour la lecture et la suppression des rôles, ou bien s'il ne vaut mieux pas carrément tout implémenter directement dans le ModelViewSet.

#53 - 22 janvier 2018 22:04 - Frédéric Péters

Commentaire de personne allant appeler l'API, ça m'irait bien que l'organizational unit ne soit pas obligatoire, qu'à défaut ça soit l'OU par défaut qui soit utilisée.

#54 - 22 janvier 2018 23:26 - Paul Marillonnet

Ok c'est noté.

#55 - 24 janvier 2018 15:56 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-API-20706.patch ajouté

Voilà, dans l'esprit, ce qui me paraît le plus direct.
En pratique, ça ne me plaît que moyennement d'avoir les permissions sur la création et la mise à jour dans le sérialisateur, et celles sur la lecture et la destruction dans le viewset.

J'essaie d'avoir un jeu de tests cohérent qui passent, et je vais voir si je déplace tout dans le viewset.

Edit: Il faut que j'abandonne cette mauvaise habitude de ne relire mes patches qu'une fois qu'ils sont sur la page du ticket. `s/kwars/kwargs/' dans mon précédent patch.

#56 - 25 janvier 2018 14:43 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-API-20706.patch ajouté

Avec quelques simplifications et ajout de tests (qui passent).

Ici on considère que la permission d'obtenir la description d'un rôle ou la liste des rôles est indépendante de toute appartenance à une OU.
En pratique, pour ModelViewSet.retrieve() et ModelViewSet.list(), on vérifie directement request.user.has_perm('a2_rbac.view_role'), et non pas la méthode similaire has_ou_perm().
C'est ce qui me paraît le plus simple et le mieux (que les permissions en lecture seules soient décorrélées de l'OU), mais je me plante peut-être.

Je voudrais encore implémenter un contrôle d'accès plus propre sur la modification des attributs d'un rôle.
En particulier la possibilité de modifier l'OU à laquelle le rôle est rattaché, qui me semble la porte ouverte à une montée en privilège (par exemple un utilisateur qui change l'OU du <role 'dummy'> valant d'abord <ou 'dummy'> pour y mettre <ou 'admin'> à la place).

#57 - 26 janvier 2018 11:10 - Paul Marillonnet

J'ai l'impression que modifier l'OU d'un rôle nécessiterait logiquement d'avoir les droits de suppression d'un rôle dans l'OU source, et ceux de création d'un rôle dans l'OU de destination. Est-ce que c'est logique ou bien exagéré ?

Edit: Ou bien complètement interdire la modification d'OU via l'API, considérant que cet attribut est figé à la création -- et donc que, pour "modifier" l'OU, l'appelant doit en fait créer un nouveau rôle dans la nouvelle OU, pour ensuite supprimer l'ancien rôle (et donc posséder les permissions de création/suppression adéquates pour chacune des OU) ?

#59 - 29 janvier 2018 12:25 - Benjamin Dauvergne

Ok je m'en occupe, tu veux que je pose ça sur la dév ?

#61 - 29 janvier 2018 13:15 - Frédéric Péters

Ce patch ne s'applique pas sur master; tu peux pusher une branche wip/, idéalement rebasée ?

#64 - 29 janvier 2018 17:07 - Benjamin Dauvergne

C'était pas un problème de rebasage, juste que le patch du renommage de l'API était nommé 0001 aussi, faut générer des séries avec git format-patch HEAD~~~ (autant de tilde que tu veux remonter dans le passé).

#65 - 29 janvier 2018 20:12 - Frédéric Péters

- Lié à Development #21488: inclure les attributs de rôle dans l'API ajouté

#66 - 15 février 2018 12:25 - Benjamin Dauvergne

Les patchs me vont, il faut juste virer le WIP et tu peux pousser.

#67 - 15 février 2018 12:40 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Avec quelques simplifications et ajout de tests (qui passent).
>
> Ici on considère que la permission d'obtenir la description d'un rôle ou la liste des rôles est indépendante de toute appartenance à une OU.
> En pratique, pour ModelViewSet.retrieve() et ModelViewSet.list(), on vérifie directement request.user.has_perm('a2_rbac.view_role'), et non pas la méthode similaire has_ou_perm().
> C'est ce qui me paraît le plus simple et le mieux (que les permissions en lecture seules soient décorrélées de l'OU), mais je me plante peut-être.
>
> Je voudrais encore implémenter un contrôle d'accès plus propre sur la modification des attributs d'un rôle.
> En particulier la possibilité de modifier l'OU à laquelle le rôle est rattaché, qui me semble la porte ouverte à une montée en privilège (par exemple un utilisateur qui change l'OU du <role 'dummy'> valant d'abord <ou 'dummy'> pour y mettre <ou 'admin'> à la place).

Et donc je reviens là dessus que j'avais ignoré, je ferai les choses comme suit:

- au lieu de exclude utiliser fields pour ne voir que uuid, name, slug, description et ou, le exclude c'est un problème pour l'avenir (et ça laisse voir trop de choses encore, on doit pas voir les membres pour l'instant)
- interdire les modifications d'ou, via if self.instance.ou != validated_data['ou']:, faire attention à partial_update (dans ce cas validated_data['ou'] peut-être None)

#68 - 15 février 2018 12:41 - Benjamin Dauvergne

Les tests ne testent pas GET sur un rôle et GET sur la liste des rôles.

#69 - 16 février 2018 11:13 - Paul Marillonnet

- Fichier 0001-add-role-creation-API-20706.patch ajouté

Benjamin Dauvergne a écrit :

> Les tests ne testent pas GET sur un rôle et GET sur la liste des rôles.

Tu veux dire que les deux tests test_api_get_role_description et test_api_get_role_list ne font pas le boulot ?

Je pose ici le patch à jour avec la désactivation de la modification de l'attribut d'OU.

#70 - 19 février 2018 11:42 - Benjamin Dauvergne

Le check_perm devrait être directement sur l'instance en fait, via user.has_perm(perm, obj=instance), je virerai simplement check_perm partout pour mettre explicitement ce qu'on souhaite faire, avec par exemple sur le serializer:

```
    @property
    def user(self):
         return self.context['request'].user

...

    if self.user.has_perm/has_ou_perm...
```

Comme ça c'est explicite à l'endroit de la décision.

#71 - 19 février 2018 11:47 - Benjamin Dauvergne

Plutôt que de tester si l'ou et de corriger de façon transparent, dans le *init* je mettrai le champ ou en readonly si self.instance existe.

```
def __init__(self, *args, **kwargs):
    super(.., self).__init__(*args, **kwargs)
    if self.instance:
        self.fields['ou'].read_only = True
```

Il faudrait un test correspondant.

Pour les tests GET j'ai du les rater désolé.

#72 - 19 février 2018 17:06 - Paul Marillonnet

- Fichier 0001-add-role-creation-API-20706.patch ajouté

Benjamin Dauvergne a écrit :

> Le check_perm devrait être directement sur l'instance en fait, via user.has_perm(perm, obj=instance), je virerai simplement check_perm partout pour mettre explicitement ce qu'on souhaite faire, avec par exemple sur le serializer:

Le code du serializer teste la permission dans l'OU dans un premier temps, et sur l'instance en particulier dans un second temps.
J'ai conservé une fonction check_perm, parce que je pense que le code peut-être factorisé dans le serializer, même si cette fonction ne fait plus appel au code du ModelViewSet associé.

> Plutôt que de tester si l'ou et de corriger de façon transparent, dans le init je mettrai le champ ou en readonly si self.instance existe.

Bien vu, merci. Corrigé dans mon patch.

> Il faudrait un test correspondant.

Les tests test_api_put_role et test_api_patch_role vérifient que l'OU est accessible en lecture uniquement. Dis-moi si tu trouves ça un peu léger, je rajouterai des tests plus poussés.

#73 - 19 février 2018 17:42 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Benjamin Dauvergne a écrit :
>
> > Le check_perm devrait être directement sur l'instance en fait, via user.has_perm(perm, obj=instance), je virerai simplement check_perm partout pour mettre explicitement ce qu'on souhaite faire, avec par exemple sur le serializer:
>
> Le code du serializer teste la permission dans l'OU dans un premier temps, et sur l'instance en particulier dans un second temps.
> J'ai conservé une fonction check_perm, parce que je pense que le code peut-être factorisé dans le serializer, même si cette fonction ne fait plus appel au code du ModelViewSet associé.

Non clairement le code est plus confus écrit comme cela, check_perm() fait trop de boulot quand il teste has_perm() + has_ou_perm(), has_ou_perm() n'est nécessaire que dans le create.

Au niveau de la vue je me rend compte que le check_perm sur list est inutile aussi, c'est qu'il faut faire c'est implémenter:

```
def get_queryset(self):
   return request.user.filter_by_perm('a2_rbac.view_role', super(RolesAPI, self).get_queryset())
```

Parce que là par exemple le listing par OU ne fonctionnerait pas (tu fais un simple check request.user.has_perm('a2_rbac.view_role') cela veut dire que seuls les utilisateurs ayant une permission globale auront accès à cette API).

Donc pas de check dans list(), retrieve() non plus (implicite via get_queryset()), je ne vois que perform_destroy() ou un request.user.has_perm('a2_rbac.delete_role', obj=instance) suffira.

> Il faudrait un test correspondant.

> Les tests test_api_put_role et test_api_patch_role vérifient que l'OU est accessible en lecture uniquement. Dis-moi si tu trouves ça un peu léger, je rajouterai des tests plus poussés.

Ça ne devrait pas lever une erreur de mettre un champ read-only ? (Mon DRF est rouillé).

#74 - 19 février 2018 18:29 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Au niveau de la vue je me rend compte que le check_perm sur list est inutile aussi, c'est qu'il faut faire c'est implémenter:
> [...]

Implémenté ça (la méthode get_queryset de la classe parente n'est apparemment pas faite pour être appelée) :

```
    def get_queryset(self):
        return self.request.user.filter_by_perm('a2_rbac.view_role', get_role_model().objects.all())
```

Mais du coup c'est des 404 (et non plus des 403) qui sont renvoyés en cas d'accès non autorisé. Pas grave ?

#75 - 19 février 2018 18:56 - Paul Marillonnet

- Fichier 0001-WIP-add-role-creation-API-20706.patch ajouté

Et donc le patch WIP avec tes remarques incluses.

#76 - 19 février 2018 21:21 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Benjamin Dauvergne a écrit :
>
> > Au niveau de la vue je me rend compte que le check_perm sur list est inutile aussi, c'est qu'il faut faire c'est implémenter:
> > [...]
>
> Implémenté ça (la méthode get_queryset de la classe parente n'est apparemment pas faite pour être appelée) :
> [...]
>
> Mais du coup c'est des 404 (et non plus des 403) qui sont renvoyés en cas d'accès non autorisé. Pas grave ?

Non c'est mieux, si tu ne dois pas voir quelque chose, autant ne pas savoir qu'il est là.

#77 - 19 février 2018 21:23 - Benjamin Dauvergne

Faut virer check_perm merci :)

#78 - 20 février 2018 17:23 - Paul Marillonnet

- Fichier 0001-add-role-creation-API-20706.patch ajouté

Benjamin Dauvergne a écrit :

> Faut virer check_perm merci :)

Hop pardon, c'est pas faute d'avoir demandé :)

#79 - 20 février 2018 18:27 - Benjamin Dauvergne

- Ça ça ne marche que parce que self.instance est à None et donc en fait ça fait un check global, donc personne pourra créer des rôles juste dans une seule OU, faut laisser que le check qui suit.

```
        # Check role-creation permissions:
        if not self.user.has_perm('a2_rbac.add_role', obj=self.instance):
            raise PermissionDenied(u'User %s can\'t create role %r' % (self.user, self.instance))
```

- ou ne peut jamais être None (ou alors il y a un problème, "required=False + default").

```
        if ou and not self.user.has_ou_perm('a2_rbac.add_role', ou):
```

- [redacted]

  pas besoin de mettre %r pour le rôle, il a une méthode *unicode*() qui marche très bien.

#80 - 20 février 2018 19:17 - Benjamin Dauvergne

Oui et donc il faudrait un test avec autre chose qu'un superadmin, par exemple avec admin_ou1.

#81 - 28 février 2018 15:21 - Paul Marillonnet

- Fichier 0001-add-role-creation-API-20706.patch ajouté

Ok d'ac, le patch à jour avec tes remarques prises en compte.

#82 - 04 mars 2018 12:27 - Frédéric Péters

- Lié à Development #22251: API de création de rôle : calcul automatique du slug quand il est absent ajouté

#83 - 05 mars 2018 11:55 - Benjamin Dauvergne

J'ai fait quelques ajustements, http://git.entrouvert.org/authentic.git/log/?h=wip/20706-API-de-creation-de-role-bda notamment ne plus utilisé le slug comme clé (c'est impossible il n'est pas unique en fait), je ne sais pas pourquoi je suis passé à coté de ça. Au passage j'ai simplifié les tests et j'ai corrigé le souci qui existait toujours avec le champ ou (en plus le test était faux, il était content que l'ou redevienne l'ou par défaut après un PUT).

Si t'es ok avec tout ça je rebase et je pousse.

#84 - 05 mars 2018 18:49 - Paul Marillonnet

;Benjamin Dauvergne a écrit :

> j'ai corrigé le souci qui existait toujours avec le champ ou (en plus le test était faux, il était content que l'ou redevienne l'ou par défaut après un PUT).

C'est ma vision des choses qui était fausse. Je croyais que c'était le comportement attendu.

> Si t'es ok avec tout ça je rebase et je pousse.

Oui, testé, ack. Merci pour les modifs.

#85 - 08 mars 2018 13:41 - Frédéric Péters

- Statut changé de Nouveau à Résolu (à déployer)

Ça a été poussé.

#86 - 13 décembre 2018 22:29 - Benjamin Dauvergne

- Statut changé de Résolu (à déployer) à Fermé

Fichiers

0001-WIP-add-role-creation-api-20706.patch	3,99 ko	04 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-api-20706.patch	5,25 ko	04 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-api-20706.patch	5,71 ko	09 janvier 2018	Paul Marillonnet
0001-rename-role-membership-API-class-pre-20706.patch	2,33 ko	12 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-api-20706.patch	4,54 ko	12 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-api-20706.patch	6,21 ko	16 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-api-20706.patch	6,21 ko	16 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-api-20706.patch	6,71 ko	18 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-API-20706.patch	7,55 ko	18 janvier 2018	Paul Marillonnet
0001-add-role-creation-API-20706.patch	3,75 ko	19 janvier 2018	Paul Marillonnet
0001-add-role-creation-API-20706.patch	3,71 ko	19 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-API-20706.patch	4,61 ko	22 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-API-20706.patch	6,86 ko	24 janvier 2018	Paul Marillonnet
0001-WIP-add-role-creation-API-20706.patch	8,8 ko	25 janvier 2018	Paul Marillonnet
0001-add-role-creation-API-20706.patch	9,49 ko	16 février 2018	Paul Marillonnet
0001-add-role-creation-API-20706.patch	9,77 ko	19 février 2018	Paul Marillonnet
0001-WIP-add-role-creation-API-20706.patch	9,38 ko	19 février 2018	Paul Marillonnet
0001-add-role-creation-API-20706.patch	9,53 ko	20 février 2018	Paul Marillonnet
0001-add-role-creation-API-20706.patch	9,81 ko	28 février 2018	Paul Marillonnet