Configurer les groupes d'accès

Concept des groupes d'accès

Avant la version 2.8 de Pod, il n'existait qu'une seule notion de groupe dans Pod dont la création et l'édition étaient accessibles dans la catégorie "Authentification et Groupes" de l'Administration.

Désormais Pod découpe les choses avec deux types de groupes :

- Les Group, toujours basés sur l'ancien modèle, dont le but est de pouvoir donner à un ensemble de personnes de droits d'accès similaires sur la plateforme.
- Les AccessGroup, dont le but est de donner accès à des contenus sur la plateforme (Vidéos, pouvoir mettre en ligne des vidéos sur des chaines, partager des dossiers...).

Configuration des groupes d'accès

Pour le CAS, le setting "USER_CAS_MAPPING_ATTRIBUTES" est maintenant doté d'un attribut "groups".

Pour le LDAP, le setting "USER_LDAP_MAPPING_ATTRIBUTES" est également doté de l'attribut "groups".

N'hésitez-pas à consulter les valeurs par défauts de ces paramètres.

Cet attribut couplé au setting "CREATE_GROUP_FROM_GROUPS" permet de créer automatiquement les groups qui seront renseignés dans le champ précisé. Si le setting "CREATE_GROUP_FROM_GROUPS" n'est pas activé, seule une association automatique sur les groupes déjà existants sera faite, mais aucun ne sera créé.

> **Attention !**
>
> Désormais, tous les groupes renseignés dans le champ "affiliations" seront également créés sous forme d'AccessGroup à la seule différence que si le setting "CREATE_GROUP_FROM_AFFILIATION" est désactivé, aucune association ne sera faite, même si les groupes existent déjà dans l'application.

Importation des groupes par un fichier

Il est possible d'importer les groupes d'accès en utilisant une commande inclue dans Pod ainsi qu'un fichier JSON que vous devez fournir.
Le fichier doit se présenter de cette manière :

```
[
{
   "code_name" : "mygroup1",
   "display_name": "My grookkoup 1j",
   "users_to_add" : [],
   "users_to_remove" : ["admin"]
},
{
   "code_name" : "mygroup2",
   "display_name" : "My group 2",
   "users_to_add" : ["login1","login2"],
   "users_to_remove" : ["login3","login4"]
},

...

]
```

Chaque groupe est identifié par son **code_name** qui va permettre de l'identifier. Le **display_name** sera un nom d'affichage et peut donc être changé par l'intermédiaire de ce fichier JSON. On peut également préciser une liste d'utilisateurs à ajouter au groupe (**users_to_add**) et une liste d'utilisateurs à retirer du groupe (**users_to_remove**). Si les utilisateurs précisés n'existent pas dans l'application, ils seront ignorés et le résultat sera loggé.

La commande à lancer pour l'importation est la suivante : **python manager.py accessgroups import_json myjson.json**

Manipulation des groupes par l'API Rest

En plus des routes classiques documentées dans /rest disponibles pour chaque modèle de Pod, le modèle AccessGroup dispose de 3 routes afin de mieux gérer ceux ci :

- /accesgroups_set_users_by_name qui prend les attributs **code_name** et **users** (même format que pour le JSON) dans une requête POST afin de pouvoir ajouter des utilisateurs dans un groupe d'accès.
- /accesgroups_remove_users_by_name qui prend les attributs **code_name** et **users** (même format que pour le JSON) dans une requête POST afin de pouvoir supprimer des utilisateurs dans un groupe d'accès.
- /accesgroups_set_groups_by_username qui prend les attributs **username** et **groups** (même format que pour le JSON) dans une requête POST afin de pouvoir affilier un utilisateur à des groupes
- /accesgroups_remove_groups_by_username qui prend les attributs **username** et **groups** (même format que pour le JSON) dans une requête POST afin de pouvoir supprimer des groupes à un utilisateur.