Publik - Bug #50850

Incohérence des modules chargés par uwsgi [était: TypeError: get_adapters() got an unexpected keyword argument 'request']

04 février 2021 11:23 - Sentry Io

Statut:	Fermé	**Début:**	04 février 2021
Priorité:	Normal	**Echéance:**	
Assigné à:	Thomas Noël	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Club:**	Non
Planning:	Non		

Description

https://sentry.entrouvert.org/entrouvert/publik/issues/28971/

```
TypeError: get_adapters() got an unexpected keyword argument 'request'
(9 additional frame(s) were not displayed)
...
  File "mellon/views.py", line 267, in authenticate
    user = auth.authenticate(saml_attributes=attributes)
  File "django/contrib/auth/__init__.py", line 70, in authenticate
    user = _authenticate_with_backend(backend, backend_path, request, credentials)
  File "django/contrib/auth/__init__.py", line 116, in _authenticate_with_backend
    return backend.authenticate(*args, **credentials)
  File "mellon/backends.py", line 37, in authenticate
    adapters = utils.get_adapters(idp, request=request)
  File "mellon/utils.py", line 172, in f
    return list(func(*args, **kwargs))
```

Demandes liées:

Lié à django-mellon - Development #50833: Stocker la requête dans l'objet ada...	**Fermé**	**03 février 2021**

Historique

#1 - 04 février 2021 11:24 - Lauréline Guérin

- Projet changé de Suivi des traces à django-mellon

#2 - 04 février 2021 11:25 - Serghei Mihai

- Lié à Development #50833: Stocker la requête dans l'objet adapter ajouté

#4 - 04 février 2021 11:48 - Serghei Mihai

Il faut recharger les services qui dépendent de mellon.

#5 - 04 février 2021 11:53 - Thomas Noël

On peut pas "corriger" plutôt que de s'obliger à ça ? (infernal lors des mises en prod)

#7 - 04 février 2021 12:05 - Benjamin Dauvergne

Je ne comprends pas comment django-mellon a pu partiellement se charger (ici on mellon/backends.py du commit en question mais mellon/utils.py d'avant commit).

#8 - 04 février 2021 12:15 - Serghei Mihai

Serghei Mihai a écrit :

> Il faut recharger les services qui dépendent de mellon.

Non, il faut chercher à comprendre pourquoi le nouveau backends.py du paquet a été pris en compte, mais pas utils.py

#10 - 04 février 2021 15:04 - Benjamin Dauvergne

Constat :

- mellon/backends.py a bien été rechargé dans le worker
- mellon/utils.py non

J'ai une hypothèse :

- on a des workers qui sont régulièrement renouvelés à partir du process maître dans lequel l'interpréteur python est déjà initialisé avec l'application wsgi (mode prefork vs lazy-apps[1])
- les backends sont chargés dynamiquement via importlib.import_module dans django.contrib.auth.authenticate(), vraisemblablement que dans le process maître aucun backend n'est chargé (idem dans get_adapters() mais on arrive pas jusque là)
- il est probable que pour une raison ou une autre mellon.utils est déjà chargé dans le process parent (du style chargement de mellon.utils par mllon.views lui même chargé par mellon.urls lors de la configuration de ROOT_URLCONF dans les settings)

Lors du fork d'un nouveau worker (après une mise à jour de django-mellon), on se retrouve avec l'ancien mellon.utils et le nouveau mellon.backends chargé dynamiquement.

Il est possible que ce comportement ait aussi été à l'origine des soucis qu'on avait eu souvent avec les plugins authentic sous forme de plugin setuptools (il me semble que gunicorn fonctionne aussi en mode prefork comme uwsgi en standard).

Je proposerais de tester lazy-apps = true sur la recette et de faire mumuse dans le code (genre virer request dans get_adapters() et le backend, restart de uwsgi, puis le remettre et voir si ça casse toujours).

[1] https://uwsgi-docs.readthedocs.io/en/latest/articles/TheArtOfGracefulReloading.html#preforking-vs-lazy-apps-vs-lazy

PS: confirmé sur un vieux post de blog http://zarnovican.github.io/2016/02/15/uwsgi-graceful-reload/ :

> This is a problem for pre-forked apps, because you may end up with old and new python modules loaded in the same process. During code deploy, worker would "inherit" (via fork()) old version of modules already loaded by master. While modules, not part of WSGI initialization, will be loaded from filesystem (new code).

#11 - 04 février 2021 15:22 - Benjamin Dauvergne

Ça aura forcément un impact sur les performances :
1. plus de mémoire utilisée
2. possiblement un ralentissement du lancement des workers, mais ça me semble plus anecdotique que le 1.

#12 - 04 février 2021 15:38 - Benjamin Dauvergne

- Projet changé de django-mellon à Publik

- Club mis à Non

#13 - 04 février 2021 15:38 - Benjamin Dauvergne

- Sujet changé de TypeError: get_adapters() got an unexpected keyword argument 'request' à Incohérence des modules chargés par uwsgi [était: TypeError: get_adapters() got an unexpected keyword argument 'request']

#14 - 08 février 2021 10:14 - Serghei Mihai

Mais est-ce que du coup il ne serait pas intéressant d'étudier la piste de redémarrage de tous les services qui ont mellon en dépendance?
De la même manière que rabbitmq est redémarré si des packages erlang sont mis à jour.

#15 - 08 février 2021 10:54 - Benjamin Dauvergne

Pour tester mon hypothèse dans un devinst, la ligne de commande de Laureline pour lancer un service avec uwsgi : #50017#note-7

#16 - 08 février 2021 10:54 - Benjamin Dauvergne

- Assigné à mis à Thomas Noël

#17 - 08 février 2021 11:05 - Benjamin Dauvergne

Serghei Mihai a écrit :

> Mais est-ce que du coup il ne serait pas intéressant d'étudier la piste de redémarrage de tous les services qui ont mellon en dépendance?
> De la même manière que rabbitmq est redémarré si des packages erlang sont mis à jour.

La différence c'est qu'erlang y a une VM unique pour tous les logiciels Erlang (il me semble). C'est un peu le principe d'Erlang.

#18 - 10 février 2021 14:49 - Thomas Noël

Testé : le mode "lazy-apps" corrige bien comme prévu, les workers renouvelés rechargent complétement l'application.

Avec ce mode, on aurait donc eu une correction "au fil de l'eau", mais pas forcément très rapide car les workers ne sont pas rapidement mis à jour, par exemple pour combo :

```
max-requests = 500
max-worker-lifetime = 7200
```

C'est à dire 3 heures au pire. C'est long... trop long : concrètement ça se terminera quand même en une commande de restart général pour nous éviter 3 heures de traces.

Au final je pense que ça ne vaut pas le coup.

#19 - 11 février 2021 14:59 - Benjamin Dauvergne

Thomas Noël a écrit :

> C'est à dire 3 heures au pire. C'est long... trop long : concrètement ça se terminera quand même en une commande de restart général pour nous éviter 3 heures de traces.

On aurait eu aucune trace, on aurait eu l'ancien ou le nouveau code, les deux fonctionnant. Je pense toujours que faute de mieux c'est la meilleure solution concernant les mises à jour des dépendances. Encore mieux ce serait que rien ne change tant qu'on ne l'a pas dit explicitement avec un restart mais ce n'est plus possible, pour des tas de raison les workers redémarrent qu'on le veuille ou non (harakiri, max-requests, etc..) et Debian ne nous permet pas d'isoler une version du code de l'application (PS: et de ses dépendances) (faut passer à Nix ou Guix pour ça).

#20 - 15 février 2021 10:50 - Frédéric Péters

Je suis perso plutôt pour vivre avec ça, il y a quand même rarement des mises à jour de django-mellon et pas vraiment d'autre module qui aurait une telle place (django-tenant-schemas mis à jour encore moins souvent que django-mellon, eopayment mais qui concerne uniquement combo).

#21 - 14 août 2022 02:07 - Thomas Noël

- Statut changé de Nouveau à Fermé

Vivons avec (c'est la mode).