Hobo - Development #34484

authentic2: limiter les threads inutiles

02 juillet 2019 11:36 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	02 juillet 2019
Priorité:	Normal	**Echéance:**	
Assigné à:	Benjamin Dauvergne	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Actuellement on lance un thread même s'il n'y a rien à provisionner.

Révisions associées

Révision e7abfc8e - 17 juillet 2019 11:51 - Benjamin Dauvergne

agent/a2: prevent useless thread launching (#34484)

Historique

#1 - 02 juillet 2019 11:39 - Benjamin Dauvergne

- Fichier 0001-agent-a2-prevent-useless-thread-launching-34484.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

#2 - 02 juillet 2019 17:47 - Nicolas Roche

Il y a quelque chose qui m'échappe.
J'ai l'impression que tu réécris :

```
<<<
not hasattr(OBJ, STRING)
---
not getattr(OBJ, STRING, None)
>>>
```

ce qui me semble équivalent, puis

```
<<<
A or B
---
A and B
>>>
```

ce qui me semble plus restrictif comme condition de sortie à la fonction provision().

#3 - 02 juillet 2019 17:58 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Il y a quelque chose qui m'échappe.
> J'ai l'impression que tu réécris :
> [...]

Non c'est clairement pas pareil:

```
x.a = {}
print not bool(hasattr(x, 'a'))
False
print not getattr(x, 'a', None)
True
```

L'idée c'est ne pas lancer le thread s'il n'y a rien à provisionner (sir .saved et .deleted sont none, vide, absents, etc.. tous les deux)

Actuellement on lance le thread sur .saved is {} et .deleted is {}, puisqu'ils sont toujours initialisés dans __enter__.

> ce qui me semble équivalent, puis
> [...]
> ce qui me semble plus restrictif comme condition de sortie à la fonction provision().

not A and not B c'est moins restrictif que not A or not B

A	B	not A and not B
V	V	F
V	F	F
F	V	F
F	F	V

A	B	not A or not B
V	V	F
V	F	V
F	V	V
F	F	V

Pour une condition de sortie si elle est plus souvent fausse, c'est moins restrictif.

#4 - 02 juillet 2019 18:40 - Nicolas Roche

Merci pour la précision pour le premier point !
Pour le second point, je demande l'intervention d'un plus chevelu que moi.

pour moi tu testes 'not A' et non pas 'A', donc ton tableau est faussé
| not A | not B | ...
ou plutôt :
A: *l'objet possède un attribut X valué comme il faut*
not A: *l'objet possède l'attribut X seulement parcequ'il est initialisé par* __enter__
(not A) or (not B): *si un des 2 attributs à une valeur bidon (alors on sort)*

et si vraiment tu veux placer le and, alors tu peux écrire : not(A and B)

#5 - 02 juillet 2019 20:12 - Benjamin Dauvergne

En fait tu m'as amené sur un sujet sans intérêt ici : le fait est que lorsque qu'on se trouve dans un with provisionning: ça appelle provisionning.__enter__() et donc ça fait self.start() et donc aussi self.local.saved/deleted = {} et donc not hasattr(self.local, 'saved') or not hasattr(self.local, 'deleted') est toujours faux; avec ma condition des fois ce sera vrai et on évitera de lancer un thread pour rien; donc oui je change la condition pour quelque chose qui dans les faits amène le changement que je souhaite (ça devient compliqué les relectures).

Pour simplifier je pourrai faire if not (self.local.saved or self.local.deleted): mais il faudrait que je crée une classe qui hérite de thread.local pour ne pas me préoccuper de la définition des deux variables (je ne connaissais pas ce fonctionnement avant, j'ai relu la doc de threading pour le voir).

#6 - 03 juillet 2019 22:47 - Nicolas Roche

- Statut changé de Solution proposée à Solution validée

Arf, merci pour ta patience Benjamin.
Dans do_provionning() on notifie sur chacun de ces deux attributs, donc normal
que tu ne sortes que seulement si aucun des deux n'est valué.

#7 - 04 juillet 2019 00:04 - Benjamin Dauvergne

- Fichier 0001-agent-a2-prevent-useless-thread-launching-34484.patch ajouté

- Statut changé de Solution validée à Solution proposée

Encore un peu de boulot, interdiff :

```
From ba5219057eb7eb9fad74fbf9db84a544da5e1e08 Mon Sep 17 00:00:00 2001
From: Benjamin Dauvergne <bdauvergne@entrouvert.com>
Date: Wed, 3 Jul 2019 09:28:34 +0200
```

```
Subject: [PATCH] provisionning wip

---
 hobo/agent/authentic2/middleware.py    |   5 +-
 hobo/agent/authentic2/provisionning.py | 166 ++++++++++++++-----------
 2 files changed, 96 insertions(+), 75 deletions(-)

diff --git a/hobo/agent/authentic2/middleware.py b/hobo/agent/authentic2/middleware.py
index c8701f3..f4af687 100644
--- a/hobo/agent/authentic2/middleware.py
+++ b/hobo/agent/authentic2/middleware.py
@@ -6,9 +6,8 @@ class ProvisionningMiddleware(object):
         provisionning.start()

     def process_exception(self, request, exception):
-        provisionning.clean()
+        provisionning.stop(provision=False)

     def process_response(self, request, response):
-        provisionning.provision()
+        provisionning.provision(provision=True, wait=False)
         return response
-
diff --git a/hobo/agent/authentic2/provisionning.py b/hobo/agent/authentic2/provisionning.py
index 417c357..e23f146 100644
--- a/hobo/agent/authentic2/provisionning.py
+++ b/hobo/agent/authentic2/provisionning.py
@@ -15,44 +15,63 @@ from authentic2.saml.models import LibertyProvider
 from authentic2.a2_rbac.models import RoleAttribute
 from authentic2.models import AttributeValue

+User = get_user_model()
+Role = get_role_model()
+OU = get_ou_model()
+RoleParenting = get_role_parenting_model()

-class Provisionning(object):
-    local = threading.local()
+logger = logging.getLogger(__name__)
+
+class Provisionning(threading.local):
+    __slots__ = ['threads']
     threads = set()

     def __init__(self):
-        self.User = get_user_model()
-        self.Role = get_role_model()
-        self.OU = get_ou_model()
-        self.RoleParenting = get_role_parenting_model()
-        self.logger = logging.getLogger(__name__)
+        self.stack = []

     def start(self):
-        self.local.saved = {}
-        self.local.deleted = {}
+        self.stack.append({
+            'saved': {},
+            'deleted': {},
+        })
+
+    def stop(self, provision=True, wait=True):
+        context = self.stack.pop()

-    def clean(self):
-        if hasattr(self.local, 'saved'):
-            del self.local.saved
-        if hasattr(self.local, 'deleted'):
-            del self.local.deleted
+        if provision:
+            self.provision(**context)
+            if wait:
+                self.wait()

-    def saved(self, *args):
```

```
-        if not hasattr(self.local, 'saved'):
+    @property
+    def saved(self):
+        if self.stack:
+            return self.stack[-1]['saved']
+        return None
+
+    @property
+    def deleted(self):
+        if self.stack:
+            return self.stack[-1]['deleted']
+        return None
+
+    def add_saved(self, *args):
+        if not self.stack:
             return

         for instance in args:
-            klass = self.User if isinstance(instance, self.User) else self.Role
-            self.local.saved.setdefault(klass, set()).add(instance)
+            klass = User if isinstance(instance, User) else Role
+            self.saved.setdefault(klass, set()).add(instance)

-    def deleted(self, *args):
-        if not hasattr(self.local, 'saved'):
+    def add_deleted(self, *args):
+        if not self.stack:
             return

         for instance in args:
-            klass = self.User if isinstance(instance, self.User) else self.Role
-            self.local.deleted.setdefault(klass, set()).add(instance)
-            self.local.saved.get(klass, set()).discard(instance)
+            klass = User if isinstance(instance, User) else Role
+            self.deleted.setdefault(klass, set()).add(instance)
+            self.saved.get(klass, set()).discard(instance)

     def resolve_ou(self, instances, ous):
         for instance in instances:
@@ -61,7 +80,7 @@ class Provisionning(object):

     def notify_users(self, ous, users, mode='provision'):
         if mode == 'provision':
-            users = (self.User.objects.filter(id__in=[u.id for u in users])
+            users = (User.objects.filter(id__in=[u.id for u in users])
                      .select_related('ou').prefetch_related('attribute_values__attribute'))
         else:
             self.resolve_ou(users, ous)
@@ -105,19 +124,19 @@ class Provisionning(object):
             # Find roles giving a superuser attribute
             # If there is any role of this kind, we do one provisionning message for each user and
             # each service.
-            roles_with_attributes = (self.Role.objects.filter(members__in=users)
+            roles_with_attributes = (Role.objects.filter(members__in=users)
                                      .parents(include_self=True)
                                      .filter(attributes__name='is_superuser')
                                      .exists())

-            all_roles = (self.Role.objects.filter(members__in=users).parents()
+            all_roles = (Role.objects.filter(members__in=users).parents()
                          .prefetch_related('attributes').distinct())
             roles = dict((r.id, r) for r in all_roles)
             user_roles = {}
             parents = {}
-            for rp in self.RoleParenting.objects.filter(child__in=all_roles):
+            for rp in RoleParenting.objects.filter(child__in=all_roles):
                 parents.setdefault(rp.child.id, []).append(rp.parent.id)
-            Through = self.Role.members.through
+            Through = Role.members.through
             for u_id, r_id in Through.objects.filter(role__members__in=users).values_list('user_id',
                                                                                           'role_id'):
                 user_roles.setdefault(u_id, set()).add(roles[r_id])
@@ -133,7 +152,7 @@ class Provisionning(object):
                 for ou, users in ous.iteritems():
                     for service, audience in self.get_audience(ou):
```

```
                        for user in users:
-                           self.logger.info(u'provisionning user %s to %s', user, audience)
+                           logger.info(u'provisionning user %s to %s', user, audience)
                            notify_agents({
                                '@type': 'provision',
                                'issuer': issuer,
@@ -149,7 +168,7 @@ class Provisionning(object):
                    audience = [a for service, a in self.get_audience(ou)]
                    if not audience:
                        continue
-                   self.logger.info(u'provisionning users %s to %s',
+                   logger.info(u'provisionning users %s to %s',
                                     u', '.join(map(unicode, users)), u', '.join(audience))
                    notify_agents({
                        '@type': 'provision',
@@ -162,9 +181,9 @@ class Provisionning(object):
                        }
                    })
        elif users:
-           audience = [audience for ou in self.OU.objects.all()
+           audience = [audience for ou in OU.objects.all()
                        for s, audience in self.get_audience(ou)]
-           self.logger.info(u'deprovisionning users %s from %s', u', '.join(map(unicode, users)),
+           logger.info(u'deprovisionning users %s from %s', u', '.join(map(unicode, users)),
                             u', '.join(audience))
            notify_agents({
                '@type': 'deprovision',
@@ -213,7 +232,7 @@ class Provisionning(object):
                ]

            audience = [entity_id for service, entity_id in self.get_audience(ou)]
-           self.logger.info(u'%sning roles %s to %s', mode, roles, audience)
+           logger.info(u'%sning roles %s to %s', mode, roles, audience)
            notify_agents({
                '@type': mode,
                'audience': audience,
@@ -229,7 +248,7 @@ class Provisionning(object):
            sent_roles = set(ou_roles) | global_roles
            helper(ou, sent_roles)

-    def provision(self):
+    def provision(self, saved, deleted):
        # Returns if:
        # - we are not in a tenant
        # - provsionning is disabled
@@ -239,27 +258,25 @@ class Provisionning(object):
            return
        if not getattr(settings, 'HOBO_ROLE_EXPORT', True):
            return
-       if (not getattr(self.local, 'saved', None)
-               and not getattr(self.local, 'deleted', None)):
+       if not (saved or deleted):
            return

-       t = threading.Thread(target=self.do_provision, kwargs={
-           'saved': getattr(self.local, 'saved', {}),
-           'deleted': getattr(self.local, 'deleted', {}),
-       })
+       t = threading.Thread(
+           target=self.do_provision,
+           kwargs={'saved': saved, 'deleted': deleted})
        t.start()
        self.threads.add(t)

    def do_provision(self, saved, deleted, thread=None):
        try:
-           ous = {ou.id: ou for ou in self.OU.objects.all()}
-           self.notify_roles(ous, saved.get(self.Role, []))
-           self.notify_roles(ous, deleted.get(self.Role, []), mode='deprovision')
-           self.notify_users(ous, saved.get(self.User, []))
-           self.notify_users(ous, deleted.get(self.User, []), mode='deprovision')
+           ous = {ou.id: ou for ou in OU.objects.all()}
+           self.notify_roles(ous, saved.get(Role, []))
+           self.notify_roles(ous, deleted.get(Role, []), mode='deprovision')
+           self.notify_users(ous, saved.get(User, []))
```

```
+            self.notify_users(ous, deleted.get(User, []), mode='deprovision')
         except Exception:
             # last step, clear everything
-            self.logger.exception(u'error in provisionning thread')
+            logger.exception(u'error in provisionning thread')
         finally:
             self.threads.discard(threading.current_thread())

@@ -271,12 +288,9 @@ class Provisionning(object):
         self.start()

     def __exit__(self, exc_type, exc_value, exc_tb):
-        if exc_type is None:
-            self.provision()
-            self.clean()
-            self.wait()
-        else:
-            self.clean()
+        if not self.stack:
+            return
+        self.stop(provision=exc_type is None)

     def get_audience(self, ou):
         if ou:
@@ -302,64 +316,72 @@ class Provisionning(object):
         return urljoin(base_url, reverse('a2-idp-saml-metadata'))

     def pre_save(self, sender, instance, raw, using, update_fields, **kwargs):
+        if not self.stack:
+            return
         # we skip new instances
         if not instance.pk:
             return
-        if not isinstance(instance, (self.User, self.Role, RoleAttribute, AttributeValue)):
+        if not isinstance(instance, (User, Role, RoleAttribute, AttributeValue)):
             return
         # ignore last_login update on login
-        if isinstance(instance, self.User) and update_fields == ['last_login']:
+        if isinstance(instance, User) and update_fields == ['last_login']:
             return
         if isinstance(instance, RoleAttribute):
             instance = instance.role
         elif isinstance(instance, AttributeValue):
-            if not isinstance(instance.owner, self.User):
+            if not isinstance(instance.owner, User):
                 return
             instance = instance.owner
-        self.saved(instance)
+        self.add_saved(instance)

     def post_save(self, sender, instance, created, raw, using, update_fields, **kwargs):
+        if not self.stack:
+            return
         # during post_save we only handle new instances
-        if isinstance(instance, self.RoleParenting):
-            self.saved(*list(instance.child.all_members()))
+        if isinstance(instance, RoleParenting):
+            self.add_saved(*list(instance.child.all_members()))
             return
         if not created:
             return
-        if not isinstance(instance, (self.User, self.Role, RoleAttribute, AttributeValue)):
+        if not isinstance(instance, (User, Role, RoleAttribute, AttributeValue)):
             return
         if isinstance(instance, RoleAttribute):
             instance = instance.role
         elif isinstance(instance, AttributeValue):
-            if not isinstance(instance.owner, self.User):
+            if not isinstance(instance.owner, User):
                 return
             instance = instance.owner
-        self.saved(instance)
+        self.add_saved(instance)

     def pre_delete(self, sender, instance, using, **kwargs):
```

```
-         if isinstance(instance, (self.User, self.Role)):
-             self.deleted(copy.copy(instance))
+         if not self.stack:
+             return
+         if isinstance(instance, (User, Role)):
+             self.add_deleted(copy.copy(instance))
          elif isinstance(instance, RoleAttribute):
              instance = instance.role
-             self.saved(instance)
+             self.add_saved(instance)
          elif isinstance(instance, AttributeValue):
-             if not isinstance(instance.owner, self.User):
+             if not isinstance(instance.owner, User):
                  return
              instance = instance.owner
-             self.saved(instance)
-         elif isinstance(instance, self.RoleParenting):
-             self.saved(*list(instance.child.all_members()))
+             self.add_saved(instance)
+         elif isinstance(instance, RoleParenting):
+             self.add_saved(*list(instance.child.all_members()))

     def m2m_changed(self, sender, instance, action, reverse, model, pk_set, using, **kwargs):
+         if not self.stack:
+             return
          if action != 'pre_clear' and action.startswith('pre_'):
              return
-         if sender is self.Role.members.through:
-             self.saved(instance)
+         if sender is Role.members.through:
+             self.add_saved(instance)
              # on a clear, pk_set is None
              for other_instance in model.objects.filter(pk__in=pk_set or []):
-                 self.saved(other_instance)
+                 self.add_saved(other_instance)
              if action == 'pre_clear':
                  # when the action is pre_clear we need to lookup the current value of the members
                  # relation, to re-provision all previously enroled users.
                  if not reverse:
                      for other_instance in instance.members.all():
-                         self.saved(other_instance)
+                         self.add_saved(other_instance)
--
2.20.1
```

#8 - 04 juillet 2019 09:37 - Benjamin Dauvergne

- Fichier 0001-agent-a2-prevent-useless-thread-launching-34484.patch ajouté

Oublié de réécrire un provisionning.provision en provisionning.stop, ça signale que le middleware n'est pas malheureusement pas testé du tout, j'ai ouvert un ticket coté django-webtest pour pouvoir avancer sur cela plus facilement, https://github.com/django-webtest/django-webtest/issues/102

#9 - 04 juillet 2019 11:14 - Benjamin Dauvergne

- Fichier 0001-agent-a2-prevent-useless-thread-launching-34484.patch ajouté

Voilà, avec de magnifiques tests et des corrections en pagaille (le test sur update_field ne marchait pas, j'ai ajouté un setting exprès pour les tests pour forcer le middleware à attendre le thread de provisionning).

#10 - 04 juillet 2019 17:42 - Nicolas Roche

Merci pour les tests !
J'ai pas mal de questions, mais je vais essayer de réviser les Thread python en même temps car j'avoue ne pas être au point sur le sujet.

- Un unique objet Provisionning est attaché au module par *apps.py*:

  ```
  engine = provisionning.Provisionning()
  provisionning.provisionning = engine
  ```

  Pourquoi utiliser les slots si on n'a qu'un seul objet ?

  ```
  class Provisionning(threading.local):
  ```

```
    __slots__ = ['threads']
```

- Pour moi on ne passe qu'une seule fois à travers les middlewares, puis on débouche éventuellement sur d'autres requêtes HTTP, mais alors elles sont gérées par un autre processus. Aussi, pourquoi avoir une gestion par pile pour les statuts (saved, deleted) ?

```
    def __init__(self):
        self.stack = []
```

Au cas où, si l'on revoit le fichier *provisionning.py* (au delà du ticket) :

- L'argument thread n'est pas utilisé.

```
def do_provision(self, saved, deleted, thread=None):
```

- Pourquoi utiliser un set pour stocker les threads ? Ne sont-ils pas supposés être tous différents ?

```
class Provisionning(threading.local):
    threads = set()
```

#11 - 04 juillet 2019 18:18 - Benjamin Dauvergne

Nicolas Roche a écrit :

> Merci pour les tests !
> J'ai pas mal de questions, mais je vais essayer de réviser les Thread python en même temps car j'avoue ne pas être au point sur le sujet.
>
> - Un unique objet Provisionning est attaché au module par *apps.py*:
>   [...]
>   Pourquoi utiliser les slots si on n'a qu'un seul objet ?
>   [...]

J'ai vu que les slots sur un objet thread local n'étaient pas thread local, alors je m'en suis servi pour ma liste de threads histoire qu'elle soit globale.

> - Pour moi on ne passe qu'une seule fois à travers les middlewares, puis on débouche éventuellement sur d'autres requêtes HTTP, mais alors elles sont gérées par un autre processus. Aussi, pourquoi avoir une gestion par pile pour les statuts (saved, deleted) ?
>   [...]

Au cas où on se retrouverait à faire (sans forcément l'avoir voulu) un with engine: dans un with engine; i.e. pour que le code soit réentrant.

Mais c'est largément perfectible ça ne prend toujours pas en compte les transactions par exemple alors que ça devrait (avec un django.db.transaction.on_commit) pour ne pas provisionner un objet qui n'est pas arrivé en base finalement.

> Au cas où, si l'on revoit le fichier *provisionning.py* (au delà du ticket) :
>
> - L'argument thread n'est pas utilisé.
>   [...]

Ouaip viré.

> - Pourquoi utiliser un set pour stocker les threads ? Ne sont-ils pas supposés être tous différents ?
>   [...]

J'aime bien les set.

#12 - 04 juillet 2019 18:22 - Benjamin Dauvergne

Aussi idéalement on ne devrait pas avoir start/stop séparé dans deux fonctions on devrait avoir un context manager faisan un try/finally mais ça arrivera seulement avec les middleware nouvelle façon de Django qui n'ont qu'une seule méthode process_request pour tout faire.

#13 - 04 juillet 2019 20:08 - Nicolas Roche

- Statut changé de Solution proposée à Solution validée

Merci pour les réponses,

pour moi c'est bon.

#14 - 04 juillet 2019 20:22 - Benjamin Dauvergne

- Statut changé de Solution validée à En cours

J'ai encore trouvé un souci dans le cas où il y a une exception (merci publik-devinst).

#15 - 04 juillet 2019 23:05 - Benjamin Dauvergne

- Fichier 0001-agent-a2-prevent-useless-thread-launching-34484.patch ajouté

- Statut changé de En cours à Solution proposée

Et là j'arrête.

#16 - 05 juillet 2019 10:20 - Nicolas Roche

- Statut changé de Solution proposée à Solution validée

Désolé, j'avais un peu joué avec et je n'ai pas vu le soucis.
(j'avoue que je n'ai pas essayé de lever l'exception)

#17 - 06 juillet 2019 11:55 - Benjamin Dauvergne

- Fichier 0001-agent-a2-prevent-useless-thread-launching-34484.patch ajouté

- Statut changé de Solution validée à Solution proposée

Django-webtest a corrigé le problème que j'ai signalé et une nouvelle release est sortie, mon monkeypatch n'est plus nécessaire.

#18 - 08 juillet 2019 09:52 - Nicolas Roche

- Statut changé de Solution proposée à Solution validée

bravo pour le patch django-webtest !

#19 - 17 juillet 2019 11:51 - Benjamin Dauvergne

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

```
commit 445cc6c9f2c2756ba2affb4bfae144f8ac9c0104
Author: Benjamin Dauvergne <bdauvergne@entrouvert.com>
Date:   Tue Jul 2 11:38:34 2019 +0200

    agent/a2: prevent useless thread launching (#34484)
```

#20 - 17 juillet 2019 14:15 - Frédéric Péters

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

Fichiers

0001-agent-a2-prevent-useless-thread-launching-34484.patch	1,34 ko	02 juillet 2019	Benjamin Dauvergne
0001-agent-a2-prevent-useless-thread-launching-34484.patch	14,5 ko	03 juillet 2019	Benjamin Dauvergne
0001-agent-a2-prevent-useless-thread-launching-34484.patch	14,5 ko	04 juillet 2019	Benjamin Dauvergne
0001-agent-a2-prevent-useless-thread-launching-34484.patch	24 ko	04 juillet 2019	Benjamin Dauvergne
0001-agent-a2-prevent-useless-thread-launching-34484.patch	24,2 ko	04 juillet 2019	Benjamin Dauvergne
0001-agent-a2-prevent-useless-thread-launching-34484.patch	23,9 ko	06 juillet 2019	Benjamin Dauvergne