Passerelle - Development #11473

Idée: sas générique

22 juin 2016 10:28 - Benjamin Dauvergne

Statut:	Nouveau	**Début:**	22 juin 2016
Priorité:	Bas	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	

Description

Réfléchissement en me rasant (ou pas): l'idée serait d'avoir un décorateur pouvant rendre l'appel asynchrone en cas d'exception et si le requêteur a ajouté un champ '$reply': 'http://example.com/reply-endpoint'.

En cas d'exception il sauve les paramètre d'URL, le body, le content-type, l'identifiant de la vue ainsi que tous ses paramètre dans un modèle lié à l'objet resource, ex.:

```
class AsyncCall(models.Model):
    create = models.DateTimeField(...)
    resource = GenericForeignKey(...)
    resource_ct_id = ...
    resource_obj_id = ...
    query_string = ...
    view_name = ....
    args = ....
    kwargs = ....
    content_type = ...
    body = models.FileField(...)
    status = ...
    log = models.TextField(...)
    response_url = models.URLField(....)
    reponse_content_type = ....
    response = models.FielField(...)
    ttl = IntegerField(...)
    delay = IntegerField(...)
    next_try = models.DateTimeField(...)
```

Et on renvoie à l'appelant la réponse {'$reply_id': <async_call.id>}, l'appelant recevra ainsi la réponse à l'URL $reply enfermé dans l'enveloppe suivante {'$reply_id': <async_call.id>, 'content_type': '...', 'b64_body': ....} (à voir si on ne gère que les réponses JSON et dans ce cas on peut directement la mettre dans un champ data.

Le statut aura les valeurs suivantes CALL_RETRY, CALL_ERROR, RESPONSE_RETRY, RESPONSE_ERROR, DONE. On commence en CALL_RETRY, une tâche de fond chercher tous les AsyncCall pour lesquels next_try est expiré et les exécute en diminuant ttl de 1, si c'est réussi on passe en RESPONSE_RETRY, et on remet le ttl à sa valeur normale (disons 13, pour un délai par défaut de 30s, ce qui donne 2,8j jusqu'à expiration des réessaies) ; si ttl est à zéro on passe en CALL_ERROR ; sinon on ajoute delay à next_try et on double delay.

Dans le statut RESPONSE_RETRY on fait les choses un peu de la mème manière.

On log tout dans le champ log (on on fait un modèle à part pour cela).

En backoffice pour chaque connecteur on a une vue des AsyncCall en cours ou terminés (nettoyage automatique tous les 90j), on doit pouvoir remettre un appel dans un statut XXX_RETRY, ou modifier le corps d'un appel ou de sa réponse (cela permet de corriger des erreurs temporaires). À chaque modification on stocke un diff dans les logs.

Goodies:

- dans le cas du JSON on se permet de reformater requête et réponses pour la lisibilité (au moins à l'affichage dans le backoffice)
- permettre d'avoir des URLs d'erreur sur un connecteur, si un AsyncCall atteint le statut erreur on envoie un mail détaillé (avec les logs, le contenu de tout) aux adresses qui y sont listées
- pouvoir spécifier ttl et délai au niveau du connecteur (donc les mettre dans BaseResource)