Passerelle - Development #21481

solis: remontée des référentiels

29 janvier 2018 17:02 - Thomas Noël

Statut:	Fermé	**Début:**	29 janvier 2018
Priorité:	Normal	**Echéance:**	
Assigné à:	Thomas Noël	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Ils obéissent tous à ce genre de format :

```
get /referentiels/trans/lieu?codePays=xxx&codeCommune=xxx&codeDepartement=xxx&...

{
  "lieux": [
    {
      "code": "string",
      "libelle": "string",
      "commune": {
        "code": "string",
        "libelle": "string"
      },
      "departement": {
        "code": "string",
        "libelle": "string"
      },
      "pays": {
        "code": "string",
        "libelle": "string"
      }
    }
  ]
}
```

Rendre l'API assez transparente, en retournant une liste où on ajoute id (code) et text (libellé) pour être compatible Publik.

Prendre en charge un "q=" qui fasse un filtre sur le libellé, avec gestion des accents

Révisions associées

Révision ba0bdaa8 - 06 février 2018 08:35 - Thomas Noël

solis: factorize request.get.json calls (#21481)

Révision 44cfbe91 - 06 février 2018 08:38 - Thomas Noël

solis: add referential endpoint (#21481)

Historique

#1 - 01 février 2018 11:58 - Thomas Noël

- Projet changé de Groupe Up à Passerelle

#2 - 05 février 2018 15:33 - Thomas Noël

- Fichier 0002-solis-factorize-request.get.json-calls-21481.patch ajouté

- Fichier 0001-solis-add-referential-endpoint-21481.patch ajouté

- Patch proposed changé de Non à Oui

0001 pour ajouter la gestion des référentiels, avec prise en charge d'un "q=" mais aussi d'un "ignore=" pour supprimer des id bizarres (parce que

beaucoup de 9999="inconnu" lors de mes tests).
0002 pour factoriser un peu la technique d'attraper du json dans leur webservice

#3 - 05 février 2018 15:35 - Benjamin Dauvergne

Je prends.

#4 - 05 février 2018 16:01 - Benjamin Dauvergne

- 0001:
  - je conseille d'utilise django.utils.http.urlencode qui fonctionne mieux en général et surtout avec de l'unicode (Django décore les paramètres)
  - il me semble que si tu as une erreur de type requests.exceptions.RequestException tu n'as rien pour en faire une APIError propre (en fait c'est vrai pour tout le module, faudrait gérer ça dans la méthode request(). C'est vrai pour tout passerelle je dirai même, je ne vois que le module iParapheur qui fasse un import depuis requests.exceptions,
  - je n'écraserai pas response avec sa valeur JSON, juste au cas où que ce soit important dans une trace, plutôt faire content = response.json()
  - je mettrai un commentaire pour dire que la valeur doit toujours être un dico avec une seule clé ex.: {"departements": [...]} comme je le vois dans les tests, j'irai même jusqu'à faire un test explicite juste avant:

    ```
    if not isinstance(content, dict) or len(content) != 1 or not istinstance(content.values()[0], list):
        raise APIError('response is not a dictionnary with only one key and whose value is a list', data={'
    json_content': content})
    ```

  - idem pour le contenu de items:

    ```
    if not all(isinstance(item, dict) and item.get('code') for item in items):
        raise APIError('items must be dictionnaries with a "code" key', data={'json_content': content})
    ```

  - je pense que tu peux te permettre d'introduire simplify dans passerelle.utils
  - je pense que ce serait plus claire ça:

    ```
    def condition(item):
        if ignore and item['id'] in ignore_ids:
            return False
        if q and q not in simplify(item['text']):
            return False
        return True
    items = filter(condition, items)
    ```

    que ça

    ```
            items = [item for item in items
                        if (not q or (q in simplify(item['text'])))
                            and (not ignore or (item['id'] not in ignore_ids))]
    ```

- À voir pour plus tard mais c'est typiquement le genre de endpoint où du cache d'environ quelques dizaines de secondes ne pourrait pas faire de mal (mais ça peut aussi ne servir à rien du tout, donc à garder sous le coude)
  0002: ben même remarques que plus haut, gérer les RequestException, à voir si il vaut mieux pas le faire au niveau de passerelle.utils.Request, voir à sous classer APIError pour un APIRequestError quitte à pouvoir les catcher si on a quelque chose de mieux à dire.

#5 - 05 février 2018 16:35 - Frédéric Péters

> je pense que tu peux te permettre d'introduire simplify dans passerelle.utils

Il y a déjà un slugify dans django.utils.text, le code peut ici faire quelque chose de subtilement différent mais si cette subtilité est pertinente pour l'ensemble de passerelle il faudra l'expliciter, qu'on sache quand utiliser l'une ou l'autre.

#6 - 05 février 2018 17:42 - Thomas Noël

- Fichier 0003-solis-factorize-request.get.json-calls-21481.patch ajouté

- Fichier 0002-solis-add-referential-endpoint-21481.patch ajouté

- Fichier 0001-misc-add-passerelle.utils.simplify.patch ajouté

Benjamin Dauvergne a écrit :

> - 0001:
>   - je conseille d'utilise django.utils.http.urlencode qui fonctionne mieux en général et surtout avec de l'unicode (Django décore les paramètres)

Exact.

- il me semble que si tu as une erreur de type requests.exceptions.RequestException tu n'as rien pour en faire une APIError propre (en fait c'est vrai pour tout le module, faudrait gérer ça dans la méthode request(). C'est vrai pour tout passerelle je dirai même, je ne vois que le module iParapheur qui fasse un import depuis requests.exceptions,

C'est le monde tordu des proxies... que faire quand le proxy n'arrive pas à faire son travail, quelle erreur renvoyer ? Pour l'instant ces exceptions qu'on n'attrape pas provoquent une 500, mais bien en JSON avec le err=1 (grâce au décorateur endpoint et jsonresponse). Que faire d'autre, quel autre code d'erreur utiliser, sachant que 200 est exclu car il veut dire "j'ai réussi mon boulot mais de l'autre côté ça déconne" ? Ici Passerelle a bien échoué dans sa mission, et si ça se trouve le crash est vraiment dans sa propre config DNS ou firewall, et donc la 500 n'est pas si anormale... Pour l'instant on vit avec ça, je n'ai pas d'autre idée. (mais cf en bas du ticket).

- je n'écraserai pas response avec sa valeur JSON, juste au cas où que ce soit important dans une trace, plutôt faire content = response.json()

Yep, bien sûr.

- je mettrai un commentaire pour dire que la valeur doit toujours être un dico avec une seule clé ex.: {"departements": [...]} comme je le vois dans les tests, j'irai même jusqu'à faire un test explicite juste avant:[...]

Impec

- idem pour le contenu de items: [...]

Merci.

- je pense que tu peux te permettre d'introduire simplify dans passerelle.utils

Fait dans un premier petit patch. Version "améliorée", largement pompée sur celle de w.c.s.

- je pense que ce serait plus claire ça: [...]
  que ça
  [...]

C'est amusant je l'ai ré-écrit ainsi en même temps que je m'auto-relisais après avoir posé le patch dans le ticket. Ça fait du bien d'utiliser ce bon vieux "filter".

- À voir pour plus tard mais c'est typiquement le genre de endpoint où du cache d'environ quelques dizaines de secondes ne pourrait pas faire de mal (mais ça peut aussi ne servir à rien du tout, donc à garder sous le coude)

J'attends de voir les usages exacts, mais oui, c'est dans les plans. On a ce qu'il faut dans self.requests (cache_duration = ...).

> 0002: ben même remarques que plus haut, gérer les RequestException, à voir si il vaut mieux pas le faire au niveau de passerelle.utils.Request, voir à sous classer APIError pour un APIRequestError quitte à pouvoir les catcher si on a quelque chose de mieux à dire.

Je pense qu'il faut faire un autre ticket sur "comment réagir en cas de pépin lors de tel ou tel request", via un ou des paramètres à envoyer au self.requests, qui déclencherait une exception proprement rattrapée par le jsonresponse, avec du log plus adéquat, genre... Mais je n'ai pas envie de faire ça ici maintenant.

#7 - 05 février 2018 17:43 - Thomas Noël

Frédéric Péters a écrit :

> je pense que tu peux te permettre d'introduire simplify dans passerelle.utils

> Il y a déjà un slugify dans django.utils.text, le code peut ici faire quelque chose de subtilement différent mais si cette subtilité est pertinente pour l'ensemble de passerelle il faudra l'expliciter, qu'on sache quand utiliser l'une ou l'autre.

Ici c'est pour aider la recherche de noms de lieux, en fait, j'avais envie de reprendre l'idée qui est dans wcs, de comparer des chaines "traduites" en ascii, avec une tentative d'intelligence sur les espace, trait d'union, apostrophe, etc.

#8 - 05 février 2018 19:19 - Benjamin Dauvergne

Thomas Noël a écrit :

- il me semble que si tu as une erreur de type requests.exceptions.RequestException tu n'as rien pour en faire une APIError propre (en fait c'est vrai pour tout le module, faudrait gérer ça dans la méthode request(). C'est vrai pour tout passerelle je dirai même, je ne vois que le module iParapheur qui fasse un import depuis requests.exceptions,

> C'est le monde tordu des proxies... que faire quand le proxy n'arrive pas à faire son travail, quelle erreur renvoyer ? Pour l'instant ces exceptions qu'on n'attrape pas provoquent une 500, mais bien en JSON avec le err=1 (grâce au décorateur endpoint et jsonresponse). Que faire d'autre, quel autre code d'erreur utiliser, sachant que 200 est exclu car il veut dire "j'ai réussi mon boulot mais de l'autre côté ça déconne" ? Ici Passerelle a bien échoué dans sa mission, et si ça se trouve le crash est vraiment dans sa propre config DNS ou firewall, et donc la 500 n'est pas si anormale... Pour l'instant on vit avec ça, je n'ai pas d'autre idée. (mais cf en bas du ticket).

Je trouve juste fatiguant de recevoir un traceback pour un Timeout ou une erreur SSL ou une erreur DNS, pour moi c'est pas la faute à passerelle si le firewall est pas ouvert ou le DNS déconne, il a fait son job, c'est le monde qui est contre lui ;)

> 0002: ben même remarques que plus haut, gérer les RequestException, à voir si il vaut mieux pas le faire au niveau de passerelle.utils.Request, voir à sous classer APIError pour un APIRequestError quitte à pouvoir les catcher si on a quelque chose de mieux à dire.

> Je pense qu'il faut faire un autre ticket sur "comment réagir en cas de pépin lors de tel ou tel request", via un ou des paramètres à envoyer au self.requests, qui déclencherait une exception proprement rattrapée par le jsonresponse, avec du log plus adéquat, genre... Mais je n'ai pas envie de faire ça ici maintenant.

Ok je veux bien que tu ouvres le tiquet puisque tu sembles avoir un avis plus tranché que le mien sur ce qui a le droit d'être une erreur propre ou pas :)

#9 - 05 février 2018 19:19 - Frédéric Péters

> Ici c'est pour aider la recherche de noms de lieux, en fait, j'avais envie de reprendre l'idée qui est dans wcs, de comparer des chaines "traduites" en ascii, avec une tentative d'intelligence sur les espace, trait d'union, apostrophe, etc.

Je préférerais vraiment garder ça dans le connecteur pour le moment.

#10 - 06 février 2018 08:20 - Thomas Noël

Benjamin Dauvergne a écrit :

> Ok je veux bien que tu ouvres le tiquet puisque tu sembles avoir un avis plus tranché que le mien sur ce qui a le droit d'être une erreur propre ou pas :)

Hop #21653, Et à relire, sans doute qu'on peut faire quelque chose de simple pour éviter le logger.exception dans jsonresponse.

#11 - 06 février 2018 08:47 - Thomas Noël

- Fichier 0002-solis-add-referential-endpoint-21481.patch ajouté

- Fichier 0001-solis-factorize-request.get.json-calls-21481.patch ajouté

Voilà, avec simplify remis dans le connecteur parce qu'effectivement c'est assez local.

#12 - 06 février 2018 15:43 - Thomas Noël

- Statut changé de Nouveau à En cours

#13 - 06 février 2018 15:43 - Benjamin Dauvergne

- Statut changé de En cours à Nouveau

Ack.

#14 - 06 février 2018 15:43 - Benjamin Dauvergne

- Statut changé de Nouveau à En cours

#15 - 06 février 2018 16:06 - Thomas Noël

- Statut changé de En cours à Résolu (à déployer)

```
commit 44cfbe918e9a7dff66c32c741eda194752cb5754
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Tue Feb 6 08:38:14 2018 +0100

    solis: add referential endpoint (#21481)

commit ba0bdaa80a749e1c33b691ac32f900c6d0cc629c
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Tue Feb 6 08:35:17 2018 +0100

    solis: factorize request.get.json calls (#21481)
```

#16 - 04 août 2018 12:29 - Benjamin Dauvergne

- Statut changé de Résolu (à déployer) à Fermé

Fichiers

0002-solis-factorize-request.get.json-calls-21481.patch	4,05 ko	05 février 2018	Thomas Noël
0001-solis-add-referential-endpoint-21481.patch	8,95 ko	05 février 2018	Thomas Noël
0003-solis-factorize-request.get.json-calls-21481.patch	4,13 ko	05 février 2018	Thomas Noël
0002-solis-add-referential-endpoint-21481.patch	9,47 ko	05 février 2018	Thomas Noël
0001-misc-add-passerelle.utils.simplify.patch	2,26 ko	05 février 2018	Thomas Noël
0002-solis-add-referential-endpoint-21481.patch	9,53 ko	06 février 2018	Thomas Noël
0001-solis-factorize-request.get.json-calls-21481.patch	3,19 ko	06 février 2018	Thomas Noël