Passerelle - Development #29314

developper le connecteur Vivaticket

21 décembre 2018 13:51 - Serghei Mihai

Statut:	Fermé	**Début:**	21 décembre 2018
Priorité:	Normal	**Echéance:**	10 janvier 2019
Assigné à:	Serghei Mihai	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	

Description

Pour les résas d'evenements à Nancy.

Fonctionnalités attendues:

- remonter les réferentiels des thèmes, catégories, salles.
- soumettre la demande de réservation

Documentation sur https://dev.entrouvert.org/projects/nancy/wiki/Vivaticket

Révisions associées

Révision 0e900739 - 18 janvier 2019 14:43 - Serghei Mihai

vivaticket: add initial connector (#29314)

Historique

#1 - 14 janvier 2019 14:32 - Serghei Mihai

- Statut changé de Nouveau à En cours

Patch poussé dans dans la branche wip/29314-vivaticket-connector.
Job jenkins: https://jenkins2.entrouvert.org/job/passerelle-wip/job/wip%252F29314-vivaticket-connector/

#2 - 14 janvier 2019 16:21 - Frédéric Péters

```
+import six
```

Utiliser django.utils.six (cf #29695).

Sauf que ce module n'est même pas utilisé. (?)

```
+            external_code = email.replace('@', '').replace('.', '').replace('_', '').replace('+', '')
```

Dans la spec je vois juste un type "string" pour ce code externe, pourquoi transformer alors que tous les caractères y passeraient ?

```
+        for key in ('id', 'email', 'datetime', 'event',
+                    'theme', 'room', 'quantity'):
+            if key not in payload or not payload[key]:
+                raise WrongParameter([key], [])
```

Il y a depuis #25590 une infra permettant de valider le JSON posté, plutôt que réinventer.

#3 - 14 janvier 2019 18:15 - Serghei Mihai

Frédéric Péters a écrit :

> Sauf que ce module n'est même pas utilisé. (?)

Viré.

> [...]
>
> Dans la spec je vois juste un type "string" pour ce code externe, pourquoi transformer alors que tous les caractères y passeraient ?

Dans la pratique ça ne passe pas. S'il y a autre chose que de l'alphanumerique le webservice renvoie une 500 avec un message "Une erreur s'est produite.". Je n'ai pas cherché plus loin.

> Il y a depuis #25590 une infra permettant de valider le JSON posté, plutôt que réinventer.

Ok, appliqué. Branche mise à jour.

#4 - 14 janvier 2019 18:21 - Frédéric Péters

> Dans la pratique ça ne passe pas. S'il y a autre chose que de l'alphanumerique le webservice renvoie une 500 avec un message "Une erreur s'est produite.". Je n'ai pas cherché plus loin.

Manque alors au moins le -. (et je zappe toutes les adresses qui contiendraient autre chose que de l'ascii). (mais peut-être alors pour être sûr quand même, faire un slugify + retrait des tirets ?)

#5 - 15 janvier 2019 10:59 - Serghei Mihai

Yep, slugify est une bonne idée.
Branche à jour.

#6 - 16 janvier 2019 12:03 - Benjamin Dauvergne

Ta branche n'est pas à jour sur master et contient un patch venant de #25815 (sur lequel j'ai fait une remarque en passant).

L'email est amené à changer, est-ce que baser le code externe des contacts VivaTicket sur celui-ci est une bonne idée ? Ce code est pour moi justement là pour palier à ce problème (et je suppose qu'on doit pouvoir récupérer un contact via ce code plutôt que par l'email). Idéalement on doit déterminer un identifiant unique pour chaque utilisateur, l'envoyer comme externalcode du contact et sur chaque utilisation du contact pour un book, mettre à jour l'email si éventuellement celui-ci a changé.

```
# pseudocode
ext_code = get_ext_code_from_payload(payload)
contact = create_or_update_contact(ext_code, email=get_email(payload))
book(contact=contact, etc...)
```

Après peut-être qu'on s'en fout de conserver la continuité des réservations en cas de changement de mail...

#7 - 16 janvier 2019 15:02 - Serghei Mihai

Benjamin Dauvergne a écrit :

> L'email est amené à changer, est-ce que baser le code externe des contacts VivaTicket sur celui-ci est une bonne idée ?

En fait pour faire une résa au nom d'un usager il faut soit le "externalId", soit le mail (comme le dit la doc). J'utilise donc mail.
Je fabrique le "externalId" uniquement pour que l'API m'autorise à créer le contact s'il n'existe pas. Ensuite je ne me sers que du mail.

> Après peut-être qu'on s'en fout de conserver la continuité des réservations en cas de changement de mail...

La "user story" (comme dit Laurent) demande d'avoir la possibilité de pouvoir réserver pour une adresse mail, sans forcément d'authentification de la part de l'usager pour pouvoir mettre à jour son mail dans les contacts Vivaticket.

#8 - 16 janvier 2019 15:59 - Benjamin Dauvergne

Serghei Mihai a écrit :

> La "user story" (comme dit Laurent) demande d'avoir la possibilité de pouvoir réserver pour une adresse mail, sans forcément d'authentification de la part de l'usager pour pouvoir mettre à jour son mail dans les contacts Vivaticket.

Comme je ne sais pas à quoi sert Vivaticket vu qu'il n'y a pas de ticket client lié à celui-ci j'aurai du mal à vous contredire complètement, néanmoins on vend un portail avec des comptes en ligne, je trouverai bien que le fonctionnement avec ce dit compte en ligne fonctionne proprement, donc faudrait pour moi une fonction get_or_create_contact() de cette forme:

```
def get_or_create_contact(email, name_id=None):
     if not email:
         raise ValueError('email is mandatory')
     unhashed_external_code = email
     if name_id:
         unhashed_external_code = name_id
```

```
    external_code = hashlib.md5(unhashed_external_code.encode('utf-8')).hexdigest()
    response = self.get('Contact/Get', externalCode=email)
    if not response.ok:
        response = self.post('Contact/Post', {
            'Email': email,
            'ExternalCode': external_code
        })
        if not response.ok:
            return None
    try:
        json_response = response.json()
    except ValueError:
        return None
    if not isinstance(json_response, dict):
        return None
    internal_code = json_response.get('internalCode')
    if not internal_code:
        return None
    if response.get('email') and response.get('email') != email:
        # update email
        updated_contact = dict(json_response, email=email)
        self.put('Contact/Put', {'contact': updated_contact})
    return {'internalCodeBuyer': post_response.json['internalCode']}
```

Je n'ai aucune idée de la tête de la réponse à Contact/Get donc certainement des choses à adapter.

Dans l'appelant vérifier la valeur retour, et éventuellement lever un APIError('unable to get or create contact').

Après je viens de lire la doc et ton code ne correspond pas du tout, par exemple pour créer un contact c'est :

```
{"key":"1234567890", "contact":{"externalCode":"45982", "civility":"M", "lastName":"Smith", "firstName":"John"
, "address1":"1 rue du bourg", "zipCode":"75010", "city":"Paris", "pays":"France", "email":"smith.john@mail.co
m"}}
```

Or toi tu envoies "Key" et le contenu du contact n'est pas dans une sous-clé mais directement au premier niveau, c'est la doc qui est une blague ou bien ?

Aussi le code post() fait systématiquement deux POST au lieu d'un, si l'API key était bonne au premier coup il faut renvoyer la première réponse; j'ai peur que ça crée plusieurs objets un jour (là si ça se trouve on s'en sort à cause du externalCode).

Un autre bug, dans book tu fais 'contact': get_or_create_contact(...) mais get_or_create_contact renvoie déjà un truc de la forme {'contact': {...}}, je ne pense pas que ce soit bon.

Je t'inciterai à écrire des functests comme Manu, pour valider que tout ça fonctionne en vrai et pas seulement dans des tests unitaires.

#9 - 17 janvier 2019 19:04 - Serghei Mihai

Branche à jour avec tes remarques et correctifs au format des données.

J'aimerais bien faire des functests mais l'API (http://testapi.tickeasy.com/QA/Nancy.Web.API/swagger/ui/index#!/) pète des 500 régulièrement avec le message

{
"Message": "Une erreur s’est produite."
}

J'ai réussi à identifier que le externalCode ne doit pas dépasser 20 caractères, sinon 500. J'ai pu créer une résa depuis un wcs local.

#10 - 18 janvier 2019 13:06 - Benjamin Dauvergne

Je relis ça.

#11 - 18 janvier 2019 13:15 - Benjamin Dauvergne

Serghei Mihai a écrit :

> Branche à jour avec tes remarques et correctifs au format des données.
>
> J'aimerais bien faire des functests mais l'API (http://testapi.tickeasy.com/QA/Nancy.Web.API/swagger/ui/index#!/) pète des 500 régulièrement avec le message

Je ne vois pas comment on peut pousser en recette un truc qui ne marche pas; impossible de faire ne serait-ce qu'une série de requêtes avec un simple booking / booking avec mise à jour d'email ?

{
"Message": "Une erreur s'est produite."
}

J'ai réussi à identifier que le externalCode ne doit pas dépasser 20 caractères, sinon 500. J'ai pu créer une résa depuis un wcs local.

Ok, un code hexadécimal à 20 caractères ça fait déjà 72 bits d'entropie c'est bien suffisant.

#12 - 18 janvier 2019 13:15 - Benjamin Dauvergne

- Statut changé de En cours à Solution validée

Les functests me rassureraient mais sinon c'est ok.

#13 - 18 janvier 2019 14:57 - Serghei Mihai

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

J'ai pu créer des boooking depuis wcs sans que ça me pété dans la gueule.

```
commit 0e9007392fa978f44f0fccc2caa7493ff95c795f (HEAD -> master, origin/master, origin/HEAD)
Author: Serghei Mihai <smihai@entrouvert.com>
Date:   Mon Dec 24 11:49:43 2018 +0100

    vivaticket: add initial connector (#29314)
```

Pour les functests je me suis créé #29890.

#14 - 18 janvier 2019 15:29 - Benjamin Dauvergne

Serghei Mihai a écrit :

> Pour les functests je me suis créé #29890.

Mouais, ça sera jamais fait :)

#15 - 18 janvier 2019 22:15 - Frédéric Péters

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

#16 - 21 janvier 2019 23:01 - Thomas Noël

Sans doute pas important : en faisant les traductions je vois qu'on a un endpoint un peu bizarrement nommé "events" :

```
    @endpoint(perm='can_access', methods=['get'], description=_('Get event categories'))
    def events(self, request):
        return self.get_setting('Settings/GetEventCategory')
```

alors que ça renvoie des catégories ?... bon bref j'ai traduis ainsi :

```
#: passerelle/apps/vivaticket/models.py:138
msgid "Get event categories"
msgstr "Récupérer les catégories d'événements"
```

Et voili voilàààà

#17 - 12 avril 2019 10:27 - Benjamin Dauvergne

- Statut changé de Solution déployée à Fermé