EOPayment - Development #49148

faire de payment_status un accesseur de la méthode sur le backend

05 décembre 2020 09:16 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	05 décembre 2020
Priorité:	Normal	**Echéance:**	
Assigné à:	Benjamin Dauvergne	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Ce sera plus simple pour détecter si la méthode est implémentée ou pas que d'utiliser NotImplementedError.

Demandes liées:

Bloque Combo - Development #49149: utiliser la méthode payment_status d'eopay...	**Fermé**	**05 décembre 2020**

Révisions associées

Révision dbb2301e - 29 janvier 2021 15:38 - Benjamin Dauvergne

misc: transform Payment.payment_status into a property (#49148)

Historique

#1 - 05 décembre 2020 09:17 - Benjamin Dauvergne

- Fichier 0001-misc-transform-Payment.payment_status-into-a-propert.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

#2 - 05 décembre 2020 09:18 - Benjamin Dauvergne

- Bloque Development #49149: utiliser la méthode payment_status d'eopayment quand elle est présente pour valider les transactions en tâche de fond ajouté

#3 - 16 décembre 2020 17:26 - Valentin Deniaud

Mouais, en plus d'être pas ouf niveau lisibilité cette approche perd la signature commune de la méthode imposée par la classe Payment, je pense qu'il ne faut pas faire comme ça.

Plutôt implémenter la méthode dans PaymentCommon histoire que tous les backends l'aient, et lui faire retourner None ou une constante par défaut.

#4 - 16 décembre 2020 18:28 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> Plutôt implémenter la méthode dans PaymentCommon histoire que tous les backends l'aient, et lui faire retourner None ou une constante par défaut.

Bon ben je vais rajouter un flag alors, ça me paraissait vraiment plus simple de faire comme cela, on fait ça partout sans avoir de signature commune à partager, ça n'a choqué personne (wcs.fields.Field.store_display_value, etc...).

#5 - 17 janvier 2021 12:03 - Benjamin Dauvergne

- Fichier 0001-misc-transform-Payment.payment_status-into-a-propert.patch ajouté

#6 - 19 janvier 2021 14:36 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> on fait ça partout sans avoir de signature commune à partager

Ce qui est spécifique ici ça me semble être que Payment est la classe « publique » de eopayment, dont le rôle est de manipuler les backends « privés ». Et donc c'est important que les trucs publics aient une signature fixe.

Valentin Deniaud a écrit :

> pas ouf niveau lisibilité

Et je me rends compte que ce qui rend cette histoire cryptique c'est dans common.py

```
199     payment_status = None
```

Tu peux peut-être faire sauter ça, en remplaçant par un truc normal genre

```
def payment_status(self, transaction_id, **kwargs):
    raise NotImplementedError
```

et retirer le raise de la méthode dans __init__.py ?

Ça empêche naturellement le raccourci que prenait le premier patch, et dans le deuxième patch has_payment_status deviendrait un truc qui rattrape l'exception. Mais ça m'aiderait probablement de voir ou tout ça va être utilisé, il y a un ticket côté lingo ?

#7 - 20 janvier 2021 09:45 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> Ça empêche naturellement le raccourci que prenait le premier patch, et dans le deuxième patch has_payment_status deviendrait un truc qui rattrape l'exception.

Tu me demandes de faire ce que ce ticket essaie d'éviter depuis le début, appeler la méthode pour savoir si elle existe.

> Mais ça m'aiderait probablement de voir ou tout ça va être utilisé, il y a un ticket côté lingo ?

#49149 le code n'y est pas encore, je dois le réécrire pour ces changements.

#8 - 20 janvier 2021 10:01 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> Valentin Deniaud a écrit :
>
> > Ça empêche naturellement le raccourci que prenait le premier patch, et dans le deuxième patch has_payment_status deviendrait un truc qui rattrape l'exception.
>
> Tu me demandes de faire ce que ce ticket essaie d'éviter depuis le début, appeler la méthode pour savoir si elle existe.

J'ai pas l'impression, je suis d'accord sur le principe du patch qui est pour moi « c'est nul d'avoir une lib qui lève des NotImplementedError qu'il faut ensuite gérer ». Mais à partir du moment où c'est géré en interne ça me paraissait OK (et davantage dans les clous que d'avoir une méthode qui des fois est une méthode, des fois est None).

Mais OK pour ne pas faire comme ça, je reprends, dans le patch :

```
244         return bool(getattr(self.backend, 'payment_status', False))
```

C'est bizarre comme ligne, sans bizarrerie ça devrait être

```
244         return hasattr(self.backend, 'payment_status')
```

Et on en revient à virer ce payment_status = None, en ne mettant rien à la place (avec petite adaptation à Payment.payment_status).

#9 - 29 janvier 2021 15:38 - Benjamin Dauvergne

- Fichier 0001-misc-transform-Payment.payment_status-into-a-propert.patch ajouté

Ok.

#10 - 29 janvier 2021 15:51 - Emmanuel Cazenave

- Statut changé de Solution proposée à Solution validée

#11 - 29 janvier 2021 16:14 - Benjamin Dauvergne

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

```
commit dbb2301eb54d27e6a1a7f0d99b4a48b083513881
Author: Benjamin Dauvergne <bdauvergne@entrouvert.com>
Date:   Sat Dec 5 09:16:30 2020 +0100

    misc: transform Payment.payment_status into a property (#49148)
```

#12 - 08 février 2021 21:16 - Frédéric Péters

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

Fichiers

0001-misc-transform-Payment.payment_status-into-a-propert.patch	1,45 ko	05 décembre 2020	Benjamin Dauvergne
0001-misc-transform-Payment.payment_status-into-a-propert.patch	1,86 ko	17 janvier 2021	Benjamin Dauvergne
0001-misc-transform-Payment.payment_status-into-a-propert.patch	2,26 ko	29 janvier 2021	Benjamin Dauvergne