Combo - Development #47513

lingo : dans BasketItemPayView prendre l'email de l'utilisateur lié à l'item si a rien d'autre

09 octobre 2020 12:12 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	09 octobre 2020
Priorité:	Normal	**Echéance:**	
Assigné à:	Benjamin Dauvergne	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Il arrivera qu'aucun email ne soit fourni ni lors de la création de l'item ni lors de l'appel à l'URL de paiement, en dernier recours on pourrait simplement prendre l'email de l'utilisateur lié à l'item (il n'y en aura pas forcément donc ce n'est pas parfait).

Révisions associées

Révision 902a5227 - 03 novembre 2020 11:13 - Benjamin Dauvergne

lingo: move email extraction to payment views (#47513)

There is too much non-linear logic around emails in handle_payment(),
it's better to locate this logic in the calling views where situations
are more constrained.

Historique

#1 - 12 octobre 2020 15:46 - Benjamin Dauvergne

- Assigné à mis à Benjamin Dauvergne

#2 - 12 octobre 2020 15:46 - Benjamin Dauvergne

- Fichier 0001-lingo-use-common-email-of-basket-item-s-users-47513.patch ajouté

- Tracker changé de Support à Development

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Le cas avec plusieurs items est théoriqes : ça n'est possible que via la cellule panier, ça tape dans PayView qui prend tous les items d'une régie pour un utilisateur authentifié, et donc on a son email (et/ou de toute façon c'est le même que les utilisateurs liés aux items).

#3 - 13 octobre 2020 15:43 - Valentin Deniaud

Sur la forme, il y a du code commenté dans le test, et les doubles sauts de lignes ça fait bizarre aussi.

Sur le fond :

- OK pour l'idée d'exploiter une info qu'on utilisait pas avant
- Mais tu écris dernier recours, or il y a encore un recours après :

```
421             if not email and item.email:
422                 kwargs['email'] = item.email
```

  C'est possible de décider que item.user.email prévaut sur item.email, mais il faudrait le dire, genre ici dans le ticket, et pourquoi. Si non, décalé le code après ces lignes pour que ce soit vraiment un dernier recourt.
- Tu vérifies qu'on ne se retrouve pas avec des emails qui diffèrent entre les items
  - Ça me paraît bien, et il faudrait peut-être étendre cette détection aux lignes ci-dessus
    - (quoique, elles font que l'email retenu est celui du dernier item du panier, ça me paraît assez ok)
  - Et tu détectes qu'il y a plusieurs emails différents, tu ne fais rien. Mais cette situation où on se retrouve à payer des items d'un autre utilisateur, genre item[0].user != item[1].user, elle est vraiment censée arriver ?

#4 - 13 octobre 2020 18:45 - Benjamin Dauvergne

- Fichier 0001-lingo-use-common-email-of-basket-item-s-users-47513.patch ajouté

Valentin Deniaud a écrit :

> Sur la forme, il y a du code commenté dans le test, et les doubles sauts de lignes ça fait bizarre aussi.

Nettoyé.

> Sur le fond :
>
> - Mais tu écris dernier recours, or il y a encore un recours après :
> [...]

Code déplacé après, j'ai complété le test pour couvrir tous les cas sur PayView et BasketItemPayView.

> - Tu vérifies qu'on ne se retrouve pas avec des emails qui diffèrent entre les items
>   - Ça me paraît bien, et il faudrait peut-être étendre cette détection aux lignes ci-dessus
>   - (quoique, elles font que l'email retenu est celui du dernier item du panier, ça me paraît assez ok)
>   - Et tu détectes qu'il y a plusieurs emails différents, tu ne fais rien. Mais cette situation où on se retrouve à payer des items d'un autre utilisateur, genre item[0].user != item[1].user, elle est vraiment censée arriver ?

Comme je le disais dans mon dernier commentaire c'est théorique, avec le code de PayView tel qu'il est ça ne peut pas arriver, si on paie plusieurs items alors on vient de PayView, et donc on est connecté, et donc on a déjà l'email de l'utilisateur connecté. Ça ne peut arriver que dans deux cas:

- on aurait plus tard une vue permettant de payer plusieurs items sans être connecté
- l'utilisateur connecté n'aurait pas d'email et les items aurait été créé attaché à son utilisateur mais avec un email différent (ça devrait être juste interdit, soit on attache un item à un utilisateur soit on fournit une email, mais pas les deux)

PS: erreur, dans le deuxième ça ne peut juste pas arriver, on a la garantit si on vient de PayView, que tous les item.user sont identiques, peut-être que item.user.email est vide mais c'est certainement identique.

#5 - 14 octobre 2020 10:06 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> Valentin Deniaud a écrit :
>
> > - Et tu détectes qu'il y a plusieurs emails différents, tu ne fais rien. Mais cette situation où on se retrouve à payer des items d'un autre utilisateur, genre item[0].user != item[1].user, elle est vraiment censée arriver ?
>
> Comme je le disais dans mon dernier commentaire c'est théorique, avec le code de PayView tel qu'il est ça ne peut pas arriver, si on paie plusieurs items alors on vient de PayView, et donc on est connecté, et donc on a déjà l'email de l'utilisateur connecté

Je m'interrogeais sur les utilisateurs liés aux items, c'est avant de penser à l'email.

> - on aurait plus tard une vue permettant de payer plusieurs items sans être connecté

Genre un lien envoyé par mail qui permettrait de payer son panier sans authentification ? Ça n'aboutit normalement pas à item[0].user != item[1].user.

> ça devrait être juste interdit

C'est là où je voulais en venir : si le cas item[0].user != item[1].user correspond à une situation qui ne doit pas arriver, il faut dire stop, et pas avoir du code qui la gère comme si c'était normal.

(mini remarque sur la forme : définir parse_qs dans le test qui l'utilise et pas comme fonction globale)

#6 - 14 octobre 2020 10:34 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> - on aurait plus tard une vue permettant de payer plusieurs items sans être connecté

> Genre un lien envoyé par mail qui permettrait de payer son panier sans authentification ? Ça n'aboutit normalement pas à item[0].user != item[1].user.

"plus tard" et "normalement" me semble incompatible, je théorise, on pourrait faire n'importe quoi dans cette vue.

> ça devrait être juste interdit

> C'est là où je voulais en venir : si le cas item[0].user != item[1].user correspond à une situation qui ne doit pas arriver, il faut dire stop, et pas avoir du code qui la gère comme si c'était normal.

Ça sort du cadre du ticket, je m'intéresse juste à l'email.

> (mini remarque sur la forme : définir parse_qs dans le test qui l'utilise et pas comme fonction globale)

Il n'y a pas d'autre usage, rien ne sert de factoriser tant qu'on a pas 3 usages.

#7 - 14 octobre 2020 10:55 - Valentin Deniaud

Benjamin Dauvergne a écrit :

> Ça sort du cadre du ticket, je m'intéresse juste à l'email.

Moi ça m'aurait été d'avoir ici un 0001 « ensure items belong to the same user » et un 0002 le patch actuel simplifié, qui prends juste item[0].user.email, mais tu peux faire un ticket séparé si tu veux.

> > (mini remarque sur la forme : définir parse_qs dans le test qui l'utilise et pas comme fonction globale)
>
> Il n'y a pas d'autre usage, rien ne sert de factoriser tant qu'on a pas 3 usages.

De quoi ? Je parle juste de déplacer ce petit helper dans le scope le plus restreint, celui de la seule fonction où il est utilisé.

Au fait, la nouvelle version du patch n'ajoute plus email dans les arguments transmis à payment.request systématiquement, or récemment tu as fais #47540 qui me paraît incompatible avec ce changement.

#8 - 14 octobre 2020 14:24 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> Moi ça m'aurait été d'avoir ici un 0001 « ensure items belong to the same user » et un 0002 le patch actuel simplifié, qui prends juste item[0].user.email, mais tu peux faire un ticket séparé si tu veux.

Je vais laisser comme ça.

> Au fait, la nouvelle version du patch n'ajoute plus email dans les arguments transmis à payment.request systématiquement, or récemment tu as fais #47540 qui me paraît incompatible avec ce changement.

J'ai compris l'inverse, voilà dans la fonction, branche à jour.

#9 - 27 octobre 2020 09:15 - Benjamin Dauvergne

- Statut changé de Solution proposée à En cours

Je vais rebaser sur #47477 quand il sera passé, c'est mieux.

#10 - 02 novembre 2020 15:38 - Benjamin Dauvergne

- Fichier 0001-lingo-use-common-email-of-basket-item-s-users-47513.patch ajouté

- Statut changé de En cours à Solution proposée

Voilà rebasé.

#11 - 02 novembre 2020 17:15 - Valentin Deniaud

- Statut changé de Solution proposée à Solution validée

Valentin Deniaud a écrit :

> Au fait, la nouvelle version du patch n'ajoute plus email dans les arguments transmis à payment.request systématiquement, or récemment tu as fais #47540 qui me paraît incompatible avec ce changement.

Tu n'avais pas répondu à ça mais si tu estimes que ça ne posera jamais problème, soit.

#12 - 02 novembre 2020 20:52 - Benjamin Dauvergne

Valentin Deniaud a écrit :

> Valentin Deniaud a écrit :
>
> > Au fait, la nouvelle version du patch n'ajoute plus email dans les arguments transmis à payment.request systématiquement, or récemment tu as fais #47540 qui me paraît incompatible avec ce changement.
>
> Tu n'avais pas répondu à ça mais si tu estimes que ça ne posera jamais problème, soit.

Tu as raison je vais énumérer les comportements

- dans PayView :
- si on est pas connecté :
  - si un paramétre de query-string nommé mail est passé dans un POST il est utilisé,
  - sinon on retourne sur la vue passée dans le paramètre 'return_url' du POST
    Ça ne concerne que des paiements en provenance de ItemView la vue publique d'une facture distante; c'est tellement spécifique que ça devrait avoir sa propre vue de paiement je pense, liée finement à ItemView, PayView resterait la vue de paiement associé uniquement à la cellule Panier, utilisable uniquement en étant connectée.
- si on est connecté (et qu'on a pas passé d'email dans un POST :/ ça n'arrive jamais mais ça reste possible), rien dans PayView mais alors dans handle_payment, l'email de l'utilisateur est utilisé et c'est tout.

- dans BasketItemPayView (qui jamais ne passe plus d'un item à handle_payment) :
- si on est connecté et qu'un email n'est pas passé dans le GET, l'email de l'utilisateur est utilisé, si un email est passé dans les paramètres du GET, cet email est utilisé
- si on est pas connecté on prend :
  - en 1er l'email dans le GET
  - ensuite l'email éventuellement attaché à l'item (item.email) lors de sa création
  - ensuite l'email commun à tous les items (et là c'est con effectivement car il n'y en a jamais plus d'un)

Je me dis qu'on pourrait en fait virer toute logique par rapport à l'email dans handle_payment(), les vues appelantes savent mieux ce qu'il faut faire.

Dans PayView :

- si connecté, email = request.user.email, fin
- si pas connecté, email = request.POST['email'] si défini, sinon revenir à la vue item_url (et tout ce bout de code devrait devenir une autre vue)

Dans BasketItemPayView:

- si connecté, email = request.user.email, fin
- si GET['email'] existe, utiliser ça (mais je pense qu'on pourrait virer ça un jour, je n'en vois pas l'usage, il n'y a que les factures qui sont éventuellement payer par un tiers avec une email différente, il faudrait forcer la condition suivante à la création d'une basketitem : soit item.user.email existe soit item.email est défini)
- si item.user existe, utiliser item.user.email,
- si item.email existe utiliser ça.

Je vais refaire mon patch dans ce sens, ça simplifiera handle_payment et mettra la logique à sa place, dans les vues.

#13 - 02 novembre 2020 22:25 - Benjamin Dauvergne

- Fichier 0001-lingo-move-email-extraction-to-payment-views-47513.patch ajouté

- Statut changé de Solution validée à Solution proposée

Au passage je découvre un autre bout de code mort et une erreur dans un test qui passe:

- dans PayView on fait request.GET.getlist('item') sauf que nul part on ne peut passer plusieurs numéros de facture, la vue ItemView ne permet d'en payer qu'une à la fois, une raison de plus de séparer ça;
- le code dans test_payment_no_basket qui crée un basketitem en passant un NameId est faux, le NameID doit être passé dans l'URL pas le contenu JSON.

#14 - 03 novembre 2020 10:26 - Valentin Deniaud

- Statut changé de Solution proposée à Solution validée

C'est cool de se retrouver avec un patch qui clarifie le code au final :)

```
         kwargs = {
-            'email': email, 'first_name': firstname, 'last_name': lastname
+            'first_name': firstname,
+            'last_name': lastname,
+            'email': email,
```

```
        }
```

Avec ce changement là en moins, c'est parfait.

#15 - 03 novembre 2020 11:13 - Benjamin Dauvergne

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

```
commit 902a52278ee5885b5e59b86e1278eb1ace633d64
Author: Benjamin Dauvergne <bdauvergne@entrouvert.com>
Date:   Mon Oct 12 15:28:30 2020 +0200

    lingo: move email extraction to payment views (#47513)

    There is too much non-linear logic around emails in handle_payment(),
    it's better to locate this logic in the calling views where situations
    are more constrained.
```

#16 - 04 novembre 2020 21:16 - Frédéric Péters

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

Fichiers

0001-lingo-use-common-email-of-basket-item-s-users-47513.patch	3,96 ko	12 octobre 2020	Benjamin Dauvergne
0001-lingo-use-common-email-of-basket-item-s-users-47513.patch	10,1 ko	13 octobre 2020	Benjamin Dauvergne
0001-lingo-use-common-email-of-basket-item-s-users-47513.patch	10,1 ko	02 novembre 2020	Benjamin Dauvergne
0001-lingo-move-email-extraction-to-payment-views-47513.patch	12 ko	02 novembre 2020	Benjamin Dauvergne