Fargo - Development #23599

Support des types de document directement dans le modèle de document

04 mai 2018 12:15 - Paul Marillonnet

Statut:	Nouveau	**Début:**	04 mai 2018
Priorité:	Normal	**Echéance:**	
Assigné à:	Paul Marillonnet	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	

Description

En l'état une telle définition de type est déjà implémentée, mais elle sert seulement à la validation des documents échangés via l'API.
Le code dans fargo.models.Validation et son sérialisateur associé dans fargo.api_views ne semble pas couvrir le besoin fonctionnel nouveau, à savoir la déclaration, par l'usager, d'un type de document lorsque cet usager attache un document lors de la saisie d'un formulaire.

Si l'idée est retenue, à voir si cela peut/doit engendrer des modifications côté w.c.s pour :

- l'option "Utiliser un fichier du porte-document"
  -> L'idée serait que, si l'usager choisit cette option, il a aussi la possibilité de voir le type du document, via un champ supplémentaire qui apparaît à titre informatif dans le formulaire.
- l'action de workflow de classement dans le porte-doc d'un document transmis via un formulaire.
  -> Le chemin inverse, à savoir, implémenter un moyen simple de transmettre via le formulaire une information relative au type de document pour ensuite stocker ce type dans fargo.
  Ça amène aussi la problématique du maintien d'une liste de types de document. En l'état, le code de fargo contient une liste settings.FARGO_DOCUMENT_TYPES, utilisée pour des problématiques de validation. Est-ce qu'on serait amené à proposer, pour l'agent ou pour l'usager, une page de gestion de ces types de document ?

Historique

#1 - 04 mai 2018 12:43 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> En l'état une telle définition de type est déjà implémentée, mais elle sert seulement à la validation des documents échangés via l'API.
> Le code dans fargo.models.Validation et son sérialisateur associé dans fargo.api_views ne semble pas couvrir le besoin fonctionnel nouveau, à savoir la déclaration, par l'usager, d'un type de document lorsque cet usager attache un document lors de la saisie d'un formulaire.
>
> Si l'idée est retenue, à voir si cela peut/doit engendrer des modifications côté w.c.s pour :
>
> - l'option "Utiliser un fichier du porte-document"
>   -> L'idée serait que, si l'usager choisit cette option, il a aussi la possibilité de voir le type du document, via un champ supplémentaire qui apparaît à titre informatif dans le formulaire.

Ok.

> - l'action de workflow de classement dans le porte-doc d'un document transmis via un formulaire.
>   -> Le chemin inverse, à savoir, implémenter un moyen simple de transmettre via le formulaire une information relative au type de document pour ensuite stocker ce type dans fargo.

Il faudrait rajouter un appel à fargo pour récupérer la liste des types de document (c'est déjà présent pour la validation) et s'en servir pour en choisir un, ensuite il faut ajouter à l'API le support de ce nouveau champ.

> Ça amène aussi la problématique du maintien d'une liste de types de document. En l'état, le code de fargo contient une liste settings.FARGO_DOCUMENT_TYPES, utilisée pour des problématiques de validation. Est-ce qu'on serait amené à proposer, pour l'agent ou pour l'usager, une page de gestion de ces types de document ?

Oui tout à fait, il faut en faire un modèle et prévoir un /manage/. Prévoir dès le départ des types indisponibles à l'utilisateur (aucun raison qu'il puisse créer une "attestation de je ne sais quoi par la mairie").

Dans l'absolu je suis très sceptique sur l'idée des types de document pour les utilisateurs, si ça sert à filtrer quelque chose. Si c'est l'utilisateur qui choisit et si plus tard ça l'empêche de sélectionner le document qu'il veut parce que connement il n'a pas choisi le bon type.

Autant quand le document vient d'une autorité administrative et qu'on voit fargo comme un passe plat entre l'autorité et elle même ou une autre autorité ça fait sens, elles se mettent d'accord sur le type on le définit, c'est déposé avec le bon type, tout va bien.

#2 - 04 mai 2018 15:47 - Paul Marillonnet

en pseudocode, les modifications dans le modèle :

```
paul@eosandbox:~/src/fargo/fargo$ git diff
diff --git a/fargo/fargo/models.py b/fargo/fargo/models.py
index a71c3cd..39ed3d3 100644
--- a/fargo/fargo/models.py
+++ b/fargo/fargo/models.py
@@ -37,12 +37,42 @@ class Origin(models.Model):
         return self.label

+class DocumentType(models.Model):
+    name = models.CharField(
+        verbose_name=_('name'),
+        max_length=512)
+    label = models.CharField(
+        verbose_name=_('label'),
+        max_length=512)
+
+
+class DocumentTypeField(models.Model):
+    document_type = models.ForeignKey(
+        DocumentType,
+        verbose_name=_('document_type'))
+    varname = models.CharField(
+        verbose_name=_('varname'),
+        max_length=512)
+    label = models.CharField(
+        verbose_name=_('label'),
+        max_length=512)
+    type = models.CharField(
+        verbose_name=_('type'),
+        max_length=512)
+    validation = models.CharField(
+        verbose_name=_('validation'),
+        max_length=512)
+
+
 class UserDocument(models.Model):
     '''Document uploaded by an user or an agent'''
     user = models.ForeignKey(
         settings.AUTH_USER_MODEL,
         verbose_name=_('user'),
         related_name='user_documents')
+    type = models.ForeignKey(
+        DocumentType,
+        verbose_name=_('type'))
     document = models.ForeignKey(
         'Document',
         related_name='user_documents',
```

L'idée c'est de reproduire la définition des types dans settings.FARGO_DOCUMENT_TYPES.
(Dans la perspective de virer cette setting, et de tout prendre en charge directement dans /manage/).

Je regarde ce que je peux faire côté API.

#3 - 04 mai 2018 15:51 - Paul Marillonnet

Je regarde aussi ce que me paraît le plus logique en termes de valeurs par défaut et cardinalité pour le pseudo-code posé ici.

#4 - 04 mai 2018 16:08 - Paul Marillonnet

Conflit de migrations lors du migrate_schemas, je regarde ce qui ne va pas.

#5 - 04 mai 2018 17:27 - Paul Marillonnet

Bon, je vois pas ce qui manque pour l'instant, je reprends plus tard.
Une fois que cette embrouille de migration est réglée, je procède comme ci-dessous, si j'ai bien compris ce que tu m'as dit Benj :
- écriture d'un sérialisateur pour les DocumentTypeField, qui inclut la phase de validation de la valeur prise par le champ.
- écriture d'un sérialisateur pour les DocumentType.
- /manage/ pour ces deux nouvelles classes

- création d'une API de liste des types
- modification de l'API de gestion des Document, pour prise en charge du type

Ok pour la non-nécessité de gestion des types par l'usager. Je vais aussi chercher à rendre obsolète FARGO_DOCUMENT_TYPES, je pense que tout faire via /manage/ sera le mieux.

#6 - 04 mai 2018 20:17 - Benjamin Dauvergne

Je suis quasiment sûr qu'on avait pris la décision de virer la validation (le code a déjà disparu de w.c.s. je pense), donc à moins de refaire un plan pour proposer de la validation avec des métadonnées je pense que tu peux te passer d'une grosse partie de test modèles.

À mon humble avis on pourrait passer sur un fonctionnement schemaless pour avancer, lors du push d'un document on accepterait un champ "metadata" contenant un dico JSON sans schéma, et dans ce cas on l'attache au document, ces données sont invisibles de l'utilisateur (il peut voir le document et normalement les deux sont liés). En cas de récupération w.c.s. (ou autre chose) récupérera aussi les métadonnées (je ne sais trop comment, pour l'instant on récupère simplement le fichier).

#7 - 07 mai 2018 09:56 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> Je suis quasiment sûr qu'on avait pris la décision de virer la validation (le code a déjà disparu de w.c.s. je pense), donc à moins de refaire un plan pour proposer de la validation avec des métadonnées je pense que tu peux te passer d'une grosse partie de test modèles.

Ok, je verrai dans un second temps pour cette validation.
Je ne crois pas que ce soit une fonctionnalité directement attendue actuellement.

> À mon humble avis on pourrait passer sur un fonctionnement schemaless pour avancer, lors du push d'un document on accepterait un champ "metadata" contenant un dico JSON sans schéma, et dans ce cas on l'attache au document, ces données sont invisibles de l'utilisateur (il peut voir le document et normalement les deux sont liés). En cas de récupération w.c.s. (ou autre chose) récupérera aussi les métadonnées (je ne sais trop comment, pour l'instant on récupère simplement le fichier).

Ok, je comprends, je vais regarder ça.

#8 - 07 mai 2018 10:13 - Mikaël Ates

Paul Marillonnet a écrit :

> Benjamin Dauvergne a écrit :
>
> > Je suis quasiment sûr qu'on avait pris la décision de virer la validation (le code a déjà disparu de w.c.s. je pense), donc à moins de refaire un plan pour proposer de la validation avec des métadonnées je pense que tu peux te passer d'une grosse partie de test modèles.
>
> Ok, je verrai dans un second temps pour cette validation.
> Je ne crois pas que ce soit une fonctionnalité directement attendue actuellement.

Si ça l'est. Exprimé par métro de Lyon et Strasbourg. C'est une validation (par un agent, au cours de l'instruction de la demande) du type de document et de méta-données qui se jointes au document.

> > À mon humble avis on pourrait passer sur un fonctionnement schemaless pour avancer, lors du push d'un document on accepterait un champ "metadata" contenant un dico JSON sans schéma, et dans ce cas on l'attache au document, ces données sont invisibles de l'utilisateur (il peut voir le document et normalement les deux sont liés). En cas de récupération w.c.s. (ou autre chose) récupérera aussi les métadonnées (je ne sais trop comment, pour l'instant on récupère simplement le fichier).
>
> Ok, je comprends, je vais regarder ça.

Ca me va de partir sans schéma pour les premiers poc. Ensuite, les concepts comme la validation du type pourrait s'inclure au schéma j'ai l'impression. Ce qui restera sans schéma c'est sûrement donc les métadonnées accolés aux documents, et leur validation, le rfr lu par l'agent dans un avis d'imposition puis accolé en MD au document par exemple.

#9 - 07 mai 2018 13:18 - Benjamin Dauvergne

Ok donc on repart sur le développement fait et livré initialement pour Alfortfille:

- en BO possibilité pour un agent de valider une pièce jointe selon un type et un schéma (métadonnée obligatoire pour le type de fichier, comme date de validité, nom, prénom, adresse, etc...)
- ensuite en FO possibilité pour l'usager de ré-utiliser la pièce la validation ayant été faite on peut sauter des étapes (les étapes de validation) dans le work-flow

Il faut bien voir que tout ce qu'on fait gagner avec ce fonctionnement c'est le temps de validation par l'agent lors de la deuxième utilisation (du point

de vue de l'usager poser un pièce avec ou sans validation ça ne change rien pour lui).

Les problématiques identifiées lors du premier développement:

- quelle nomenclature pour les types de document ?
  - soit identifier des types de document en fonction de leur usage (justificatif de domicile, de revenus, etc...) et donc un même document peut recouvrer plusieurs usages et donc plusieurs validation (un avis d'imposition ça sert au deux, une facture EDF uniquement de justificatif de domicile).
  - soit identifier les types de document par source (avis d'imposition, facture pouvant servir de justificatif, etc..) et alors pour un usage donnée on doit accepter plusieurs types de documents

- peut-on pré-remplir automatiquement un champ justificatif dans un formulaire sans action de l'usager (avec ses métadonnées) ? On le fait pour l'état civil, l'adresse déclarative ou le mail (on l'avait fait pour Alfortville)

- quid de l'accès au guichet (ou au télphone, en multi-canal) au porte-feuille des documents validés ? Si je viens au guichet et je n'ai pas le document sur moi, et j'ai besoin d'un justificatif de domicile qui est dans mon porte doc comment s'en servir ? (on pourrait voir du coté des notifications SMS/mail pour demander l'autorisation)

Pour historique on avait développé tout ça Alfortville, livré, testé par eux puis ils nous ont dit que ce n'est pas ça qu'il voulait, ça n'a jamais été mis en prod.