w.c.s. - Development #47704

faire que AttachmentEvolutionPart gère les PicklableUpload avec un storage distant

15 octobre 2020 02:58 - Thomas Noël

Statut:	Fermé	**Début:**	15 octobre 2020
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

actuellement AttachmentEvolutionPart fait une "copie" de l'objet Upload qu'elle reçoit, y compris son contenu... quand il s'agit d'un RemoteOpaqueUploadStorage ça ne marche pas (le contenu n'est pas là).

Révisions associées

Révision 9d0ca1c0 - 16 octobre 2020 11:12 - Thomas Noël

stick attachments links to backoffice when shown on backoffice (#47704)

Révision 015af975 - 23 octobre 2020 00:23 - Thomas Noël

misc: accept non-local storage as attachments in evolution (#47704)

Historique

#3 - 15 octobre 2020 12:52 - Thomas Noël

- Fichier 0001-misc-accept-non-local-storage-as-attachments-in-evol.patch ajouté

- Statut changé de Nouveau à En cours

- Patch proposed changé de Non à Oui

Travail en cours sur lequel je veux bien un premier avis. Ca passe les tests actuels et ça fonctionne "en live", je dois maintenant écrire des tests couvrant ce nouveau code s'il semble correct.

- wcs/qommon/upload_storage.py : gestion des fichiers qui n'existent pas en réalité, get_file_pointer renverra None
- wcs/wf/attachment.py : correction pour faire que les liens vers les attachments en backoffice restent bien en backoffice (dans l'objectif de déterminer la visibilité de la redirection)
- wcs/workflows.py :
  - adaptation de AttachmentEvolutionPart pour ressembler encore plus à un PicklableUpload et donc intégrer les storage=xxx. La partie délicate est le *getstate* qui intègre le calcul d'un filename pour les fichiers normaux à partir d'un digest du contenu. Ce filename est ensuite la clé qui permet de retrouver l'attachment. Ici on n'a pas de contenu donc je prends un uuid. Peut-être qu'il faudrait solidifier en prefixant avec "uuid-" afin de pouvoir mieux distinguer les deux.
  - dans convert_attachments_to_uploads on ne cherche plus à convertir les upload qui sont déjà des PicklableUpload, sinon on perd le storage= (j'ai tenté de modifier FileField.convert_value_from_anything à la place mais sans succès, c'est délicat car on doit y renvoyer un nouvel objet)

#4 - 15 octobre 2020 15:59 - Benjamin Dauvergne

Thomas Noël a écrit :

> - wcs/qommon/upload_storage.py : gestion des fichiers qui n'existent pas en réalité, get_file_pointer renverra None

Ok.

> - wcs/wf/attachment.py : correction pour faire que les liens vers les attachments en backoffice restent bien en backoffice (dans l'objectif de déterminer la visibilité de la redirection)

Ça irait dans un commit à part, je vois bien le lien, mais pour moi ça résout un problème plus général.

> - wcs/workflows.py :
> - adaptation de AttachmentEvolutionPart pour ressembler encore plus à un PicklableUpload et donc intégrer les storage=xxx. La partie délicate est le *getstate* qui intègre le calcul d'un filename pour les fichiers normaux à partir d'un digest du contenu. Ce filename est ensuite la clé qui permet de retrouver l'attachment. Ici on n'a pas de contenu donc je prends un uuid. Peut-être qu'il faudrait solidifier en prefixant avec "uuid-" afin de pouvoir mieux distinguer les deux.

Sur content alors qu'il n'y a pas de contenu je ferai un AttributeError mais ça se discute, il me semble qu'il sera caché dans un template et dans du python des AttributeError on en fait déjà plein.

- dans convert_attachments_to_uploads on ne cherche plus à convertir les upload qui sont déjà des PicklableUpload, sinon on perd le storage= (j'ai tenté de modifier FileField.convert_value_from_anything à la place mais sans succès, c'est délicat car on doit y renvoyer un nouvel objet)

J'ai essayé de comprendre si convertir des PicklableUpload avait un intérêt et je n'en vois pas, on va à chaque fois calculer le digest, trouver un fichier avec nom et le réécrire, je dirai que ça ne change rien donc (sauf pour le storage=). Peut-être qu'on peut ajouter le cas directement à convert_value_from_anything() et garder la simplicité ici ? Je ne vois que deux autres usages dans evalutils de FileField.convert_value_from_anything() qui ne seront pas impactés je pense.

À part ces remarques anecdotique ça m'a l'air de tenir la route et de couvrir tous les points concernés.

#5 - 15 octobre 2020 16:15 - Thomas Noël

Benjamin Dauvergne a écrit :

- wcs/wf/attachment.py : correction pour faire que les liens vers les attachments en backoffice restent bien en backoffice (dans l'objectif de déterminer la visibilité de la redirection)

> Ça irait dans un commit à part, je vois bien le lien, mais pour moi ça résout un problème plus général.

Yep bonne idée : #47740

- wcs/workflows.py :
  - adaptation de AttachmentEvolutionPart pour ressembler encore plus à un PicklableUpload et donc intégrer les storage=xxx. La partie délicate est le *getstate* qui intègre le calcul d'un filename pour les fichiers normaux à partir d'un digest du contenu. Ce filename est ensuite la clé qui permet de retrouver l'attachment. Ici on n'a pas de contenu donc je prends un uuid. Peut-être qu'il faudrait solidifier en prefixant avec "uuid-" afin de pouvoir mieux distinguer les deux.

> Sur content alors qu'il n'y a pas de contenu je ferai un AttributeError mais ça se discute, il me semble qu'il sera caché dans un template et dans du python des AttributeError on en fait déjà plein.

En fait je suis en train de me dire que ce content va crasher en cas de RemoteOpaque, parce que le get_file_pointer de AttachmentEvolutionPart c'est juste « return open(self.filename, 'rb') ». Je vais creuser un peu.

- dans convert_attachments_to_uploads on ne cherche plus à convertir les upload qui sont déjà des PicklableUpload, sinon on perd le storage= (j'ai tenté de modifier FileField.convert_value_from_anything à la place mais sans succès, c'est délicat car on doit y renvoyer un nouvel objet)

> J'ai essayé de comprendre si convertir des PicklableUpload avait un intérêt et je n'en vois pas, on va à chaque fois calculer le digest, trouver un fichier avec nom et le réécrire, je dirai que ça ne change rien donc (sauf pour le storage=). Peut-être qu'on peut ajouter le cas directement à convert_value_from_anything() et garder la simplicité ici ? Je ne vois que deux autres usages dans evalutils de FileField.convert_value_from_anything() qui ne seront pas impactés je pense.

« Peut-être qu'on peut ajouter le cas directement à convert_value_from_anything() » : j'ai tenté et des tests crashent, sur des choses qui semblent assez éloignées. Je suspecte que parfois ça pickle en avance alors que le contenu n'est pas encore là, ou alors que le retour de convert_value_from_anything() ne peut pas être l'objet lui-même, ou alors que les tests sont un peu trop brutaux... j'avoue que je n'ai pas creusé plus que ça, je vais regarder un peu plus.

#6 - 16 octobre 2020 17:12 - Thomas Noël

- Statut changé de En cours à Solution proposée

Thomas Noël a écrit :

> Sur content alors qu'il n'y a pas de contenu je ferai un AttributeError mais ça se discute, il me semble qu'il sera caché dans un template et dans du python des AttributeError on en fait déjà plein.

> En fait je suis en train de me dire que ce content va crasher en cas de RemoteOpaque, parce que le get_file_pointer de AttachmentEvolutionPart c'est juste « return open(self.filename, 'rb') ». Je vais creuser un peu.

Dans cette nouvelle version, une petite modif : le "filename" qui sert d'identifiant est préfixé "uuid-" s'il est généré alors qu'il n'y a pas de contenu. Ca permet de savoir que le filename n'est pas un vrai filename, et donc get_file_pointer va renvoyer None dans ce cas, et "content" va renvoyer du vide.

Je n'ai pas fait de AttributeError car l'attribut existe bien et que je voulais suivre ce qui est fait dans PicklableUpload.

> « Peut-être qu'on peut ajouter le cas directement à convert_value_from_anything() » : j'ai tenté et des tests crashent, sur des choses qui semblent assez éloignées. Je suspecte que parfois ça pickle en avance alors que le contenu n'est pas encore là, ou alors que le retour de convert_value_from_anything() ne peut pas être l'objet lui-même, ou alors que les tests sont un peu trop brutaux... j'avoue que je n'ai pas creusé plus que ça, je vais regarder un peu plus.

De ce côté là je n'ai pas réussi et finalement je n'ai pas trop envie de toucher à convert_value_from_anything, c'est un truc utilisé lors d'attribution et je ne suis pas sur qu'on teste tout ce que les gens en ont fait (par ex. les =utils.attachment(...))

Voici donc une version avec le test qui couvre en vérifiant que les AttachmentEvolutionPart répondent bien avec des fichiers RemoteOpaque.

#7 - 16 octobre 2020 17:12 - Thomas Noël

- Fichier 0001-misc-accept-non-local-storage-as-attachments-in-evol.patch ajouté

Thomas Noël a écrit :

> Voici donc une version avec le test qui couvre en vérifiant que les AttachmentEvolutionPart répondent bien avec des fichiers RemoteOpaque.

#8 - 19 octobre 2020 15:29 - Frédéric Péters

- Statut changé de Solution proposée à Solution validée

Ok ok, cycle prochain,

```
-        # there is not filename, or it was a temporary one: create it
+        # there is no filename, or it was a temporary one: create it
```

(je me vois très bien dans cinq ans faire des tickets pour virer tout ça...)

#9 - 23 octobre 2020 00:24 - Thomas Noël

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

```
commit 015af97502ea6311939b0cd72e1a4a76d4598530
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Thu Oct 15 15:45:18 2020 +0200

    misc: accept non-local storage as attachments in evolution (#47704)
```

#10 - 23 octobre 2020 01:17 - Frédéric Péters

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

Fichiers

0001-misc-accept-non-local-storage-as-attachments-in-evol.patch	7,39 ko	15 octobre 2020	Thomas Noël
0001-misc-accept-non-local-storage-as-attachments-in-evol.patch	10,7 ko	16 octobre 2020	Thomas Noël