w.c.s. - Development #41847

modifier une donnée de traitement de type fichier

17 avril 2020 18:25 - Frédéric Péters

Statut:	Fermé	**Début:**	17 avril 2020
Priorité:	Normal	**Echéance:**	
Assigné à:	Frédéric Péters	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Plutôt que nécessairement du Python moche façon vars().get('reponse_var_pj1_raw'), permettre du django.

On ne peut sans doute pas modifier la sortie de {{reponse_var_pj1_raw}} mais donc, définir un autre suffixe, genre _file, qui produirait sous forme de chaine "MARQUEUR un truc significatif permettant de retrouver le fichier MARQUEUR".

Pour le marqueur, http://www.unicode.org/faq/private_use.html, on peut juste commencer au premier, \ue000.

Demandes liées:

Lié à w.c.s. - Development #48436: Pré-remplissage d'un bloc de champ par cré...	**Fermé**	**10 novembre 2020**
Lié à w.c.s. - Development #49726: Permettre la complétion d'un bloc de champ...	**Fermé**	**23 décembre 2020**

Révisions associées

Révision 8c551089 - 21 décembre 2020 18:52 - Frédéric Péters

general: allow assigning complex types from rendered templates (#41847)

Historique

#1 - 17 avril 2020 21:08 - Thomas Noël

Ca pourrait être « \ue000file: CONTENU \ue000 » dans l'idée que le "file:" indique que c'est un fichier, et qu'un jour on aurait d'autres types "complexes" à jouer. En gros, on enverrait ce qui est dans le \ue000 à un désérialiseur générique.

(et truc rigolo, on pourrait même permettre ce \ue000 dans le résultat de gabarit de mail, et en extraire alors les fichiers à attacher, un peu à la manière de ce qu'on fait pour les boutons d'actions, bref)

#2 - 17 avril 2020 21:44 - Benjamin Dauvergne

Et sinon accepter reponse_var_pj1 et simplement ajuster FileField.convert_value_from_anything() pour vérifier que la valeur à une méthode raw() et qu'elle renvoie un fichier ?

Vous me faites peur des fois.

#3 - 17 avril 2020 21:49 - Benjamin Dauvergne

Il y aussi la possibilité de proposer directement dans un sélecteur les champs fichiers et les varname des appels de WS avec un response_type attachment (en plus je ne comprends pas bien le bins vu qu'il y a déjà une option pour stocker directement dans un champs BO sur les appels WS).

#4 - 17 avril 2020 22:37 - Frédéric Péters

La raison reste toujorus qu'il existe des endroits où c'est compliqué de fournir des listes, genre maintenant tu peux spécifier les fichiers attachés à un modèle de courriel, et le modèle est indépendant des workflows et des formulaires.

Dedans, pouvoir écrire par exemple {% firstof reponse_var_pj1_file reponse_var_pj2_file %}, ça permet un truc déjà compris par ailleurs, parce que les gabarits Django sont utilisés à quantité d'endroits.

#5 - 18 avril 2020 05:44 - Pierre Cros

Point de vue d'un fonctionnel pour ce que ça vaut : des gabarits django partout, merci.

On a fait l'effort d'apprendre leur fonctionnement, dans une certaine mesure et en mode monkey, on peut même progresser, pourquoi pas - un cours sur les boucles serait plutôt utile, par exemple.

Mais tous les endroits où on ne peut pas les utiliser deviennent des pain in the arse, des trucs sur lesquels on peut perdre des heures (en réalisation ou en support). Il faudrait d'ailleurs essayer de les lister, tiens. C'est pas que Python ou autre soit plus compliqué, c'est juste l'hétérogénéité qui est source de complexité.

On a fait le choix des gabarits - fondé ou pas j'en sais foutre rien - on doit le systématiser.

#7 - 23 juin 2020 15:53 - Serghei Mihai

L'action "Création d'une demande", avec des mappings vers des champs fichier, est concernée aussi.

#8 - 10 novembre 2020 11:52 - Frédéric Péters

- Lié à Development #48436: Pré-remplissage d'un bloc de champ par création de demande ajouté

#9 - 29 novembre 2020 19:17 - Frédéric Péters

- Assigné à mis à Frédéric Péters

#10 - 04 décembre 2020 16:55 - Frédéric Péters

- Fichier 0001-general-allow-assigning-complex-types-from-rendered-.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Quand une valeur lazy est demandée dans un gabarit (_raw, ou __str__),

```
     @property
     def raw(self):
         if self._field.store_display_value or self._field.key in ('file', 'date'):
-            return self._data.get(self._field.id)
+            raw_value = self._data.get(self._field.id)
+            return get_publisher().cache_complex_data(raw_value)
         raise AttributeError('raw')
...
     def __str__(self):
         value = self.get_value()
         if not isinstance(value, six.string_types):
-            value = str(value)
+            value = get_publisher().cache_complex_data(value)
         return force_str(value)
```

On la passe dans cache_complex_data() qui garde un dictionnaire de correspondance entre la représentation textuelle de la valeur et celle-ci,

```
+        # Keep a temporary cache of assocations between a complex data value
+        # (value) and a string reprensentation (str(value) or the dedicated
+        # str_value parameter).
```

Pour assurer que deux fichiers différents qui seraient représentés pareils soient confondus (parce que la représentation d'un fichier est son nom), on ajoute à la chaine une marque unique ("caractère" unicode privé),

```
+        # It ensures string values are unique by appending a private unicode
+        # code point, that will be removed in wcs/qommon/template.py.
...
+        str_value += chr(0xE000 + len(self.complex_data_cache))
```

Au moment d'utiliser la valeur (actions données de traitement ou création d'une demande, citées dans ce ticket), on regarde dans le dictionnaire et on prend la valeur qui peut y être,

```
+                    if formdef_field.allow_complex:
+                        complex_value = get_publisher().get_cached_complex_data(new_value)
+                        if complex_value:
+                            new_value = complex_value
```

Pour les endroits qui ne gèrent pas les données complexes, ces caractères sont juste virés,

```
+        return re.sub(r'[\uE000-\uF8FF]', '', rendered)
```

Les tests couvrent les deux actions citées et comme types de champs les fichiers, listes à choix multiples et blocs de champs (bout qui corrigera #48436).

#12 - 21 décembre 2020 16:58 - Thomas Noël

- Statut changé de Solution proposée à Solution validée

Tout ça me semble tout bon.

Deux mini choses que je me note dans un coin :

- il faudra pousser l'affaire ensuite afin qu'un pre-fill avec {{ cards|...|getlist:... }} fonctionne sur un champ liste à choix multiple
- et avoir dans le debug de l'inspecteur une information concernant le résultat d'un {{form_var_file}} en tant que donnée complexe (pour "montrer" que ça renvoie bien un fichier quand c'est utilisé en mode donnée de traitement et autres).

#13 - 21 décembre 2020 18:52 - Frédéric Péters

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

```
commit 8c551089165e62ef8ac494fa10aee7d8d0c943c8
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Sun Nov 29 17:50:37 2020 +0100

    general: allow assigning complex types from rendered templates (#41847)
```

#14 - 21 décembre 2020 20:16 - Frédéric Péters

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

#16 - 23 décembre 2020 16:48 - Frédéric Péters

- Lié à Development #49726: Permettre la complétion d'un bloc de champ depuis une action "création d'une fiche" ou "modification d'une fiche" ajouté

Fichiers

0001-general-allow-assigning-complex-types-from-rendered-.patch	22,4 ko	04 décembre 2020	Frédéric Péters