w.c.s. - Development #52110

champ de type "donnée calculée"

16 mars 2021 16:58 - Frédéric Péters

Statut:	Fermé	**Début:**	16 mars 2021
Priorité:	Normal	**Echéance:**	07 mai 2021
Assigné à:	Frédéric Péters	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

cf #27429#note-50 pour l'origin story; la description qui en est sortie est :

> Ce nouveau type de champ, dans un formulaire ou une fiche, permet d'avoir un champ caché à l'utilisateur (= n'est pas affiché dans le formulaire) dans lequel on stocke une valeur (comme une donnée de traitement dans les workflows) : cette valeur peut être une donnée de session (session_var_...), un paramètre passé dans l'url (query string), un calcul (form_var_truc|add:form_var_machin), un gabarit ({{form_var_bidule}}) ou toute autre valeur fixe.
>
> Ce champ peut être réutilisé dans la suite du formulaire et dans le workflow (pour faire des conditions, des calculs...).
>
> Il disposera d'une option qui permettra, en cas de modification du formulaire, de conserver la valeur initiale (utile pour une variable de session ou un query string) ou de recalculer la valeur (utile pour un calcul qui se fait en fonction de champs saisis plus haut).
>
> Ce champ n'apparait pas dans le rendu du formulaire (remplace le hack de mettre le style hidden sur un champ).
>
> Ce type de champ sera disponible dans les modèles de fiche et les formulaires, mais ne sera pas utilisable dans les données de traitement, les variables de workflow, les formulaires de workflow ni les blocs de champs.

Demandes liées:

Lié à Publik - Development #27429: Permettre les champs lecture seule / caché...	**Fermé**	**19 octobre 2018**

Révisions associées

Révision f78d10fd - 30 avril 2021 12:22 - Frédéric Péters

general: introduce a new "computed data" field (#52110)

Historique

#1 - 06 avril 2021 08:36 - Frédéric Péters

- Lié à Development #27429: Permettre les champs lecture seule / caché / etc. ajouté

#2 - 26 avril 2021 15:36 - Frédéric Péters

- Fichier 0001-general-introduce-a-new-computed-data-field-52110.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Voilà ça m'a pris du temps à le proposer; à noter :

- c'est un nouveau type de champ, _('Computed Data'), si on veut encore discuter du nom, on peut;
- en terme de paramètres, on y a :
  - un libellé (label), pas formellement utilisé mais pratique pour y faire référence;
  - un gabarit de calcul de la valeur ("value template"),
  - une case à cocher "Freeze on initial value", genre "figer sur la première valeur calculée", ou autre, ici pour le coup la discussion sur le libellé est sans doute une discussion utile.
- le comportement par défaut est d'avoir la valeur recalculée, par exemple avant l'évaluation des conditions de sortie, cette case à cocher, cf discussions précédentes, permet de désactiver ça, de rester sur la première valeur, en pratique c'est utile pour une première valeur qui serait {{request.GET.quelquechose}}, que la valeur ne soit pas perdue à la validation de la page, ou après avoir chargé un brouillon.
  - c'est vraiment le seul truc fonctionnellement un peu complexe.
- techniquement par contre, le truc complexe est bien sûr dans la gestion de ces données lors de la phase de saisie (wcs/forms/root.py);

    - lors de la saisie on transpore un jeton (magictoken) qui permet d'accéder aux données saisies,
    - le patch étant ça mais ne mélange pas ces données calculées dans les autres, plutôt il garde en session un dictionnaire spécifique (session.add_magictoken('%s-computed' % magictoken, computed_values)).
    - oui ça pourrait indiquer que toute la partie saisie devrait être reprise, mais pas encore maintenant.
- côté stockage ça tape par champ une colonne jsonb, ça veut dire que les données doivent être sérialisables json, j'ai quand même préféré ça à une colonne bytea avec du pickle dedans, en perpective d'évolutions / requêtes.
    - ce n'est pas directement la donnée mais un dictionnaire {'data': value, '@type': 'computed-data'} qui est stocké, c'est d'abord fait ainsi pour avoir tout le temps un dictionnaire, y compris quand value serait une chaine de caractères, pour ne pas avoir à gérer un CAST postgresql à un endroit pas prévu pour.
- ça déplace un peu de code dans variables.py, pour que le code de "inspect_keys" soit partagé avec celui des objets _structured.
- les tests couvrent les différentes situations que j'imaginais, gel ou pas de la valeur initiale, rappel de brouillon, stockagde de données complexes, utilisation en condition de sortie, etc.

Honnêtement la galère c'est la partie wcs/forms/root.py, je pense qu'il faut juste y faire confiance aux tests.

#4 - 27 avril 2021 09:47 - Lauréline Guérin

J'ai compris les tests, un peu moins le reste :)
C'est ok pour moi, mais si une autre personne veut bien relire et valider je préfère.
Si personne ne s'y colle je validerai :)

#5 - 27 avril 2021 10:37 - Thomas Noël

Je vais faire une passe aussi (dans la journée) [edit : et non, je n'ai pas encore trouvé le temps...]

#6 - 28 avril 2021 17:05 - Thomas Noël

J'ai un problème avec le save de ComputedField où on voit :

```
        if not self.varname:
            self.varname = misc.simplify(self.label, space='_')
```

la création d'identifiant en doublon me semble être un truc qui va se produire souvent. Mais je vois aussi qu'on a un required=True dans le form... et je me demande donc ce morceau de code dans le save() n'est pas juste là "pour se rassurer" ...?

Sur :

```
value = WorkflowStatusItem.compute(field.value_template, raises=True, allow_complex=True)
```

on va stocker "n'importe quoi" dans un champ calculé. C'est bien, c'est souple, mais peut-être trop : je m'interroge sur la compréhension par nos utilisateurs. Et m'inquiète un peu des objets bizarres qu'on pourrait se retrouver stocker, genre des résultats de requêtes...

Idéalement j'aurai préféré que dans ComputedField on demande de défniir un "target_type" parmi string/text/bool/file/date. Et après le compute on ferait un convert_value_from_anything

Bien sûr ça limiterait peut-être quelques usages, mais la magie et le support s'opposent.

Pour le reste, ça m'a l'air pas mal. Je dois cependant essayer de comprendre encore comme ça s'imbrique avec la notion de "live" (si tu l'as envisagé ou pas)

#7 - 28 avril 2021 17:33 - Frédéric Péters

> la création d'identifiant en doublon me semble être un truc qui va se produire souvent

Parce que souvent les gens vont mettre le même libellé ? (Il n'y a de toute façon pas d'interdiction d'identifiant en doublon, de manière générale sur les champs.)

Ça rejoint la demande de taper automatiquement un identifiant sur les champs (#41324) mais comme ici le nom de variable est obligatoire, c'est nécessaire.

Ta préférence serait à ce qu'il y ait regard sur les champs existants pour suffixer d'un compteur si jamais l'identifiant se trouvait utilisé ?

~~

> Idéalement j'aurai préféré que dans ComputedField on demande de défniir un "target_type"

J'imaginais pas exemple qu'un champ pourrait avoir comme contenu le résultat d'un appel web, qui ne rentrerait pas dans ces cases.

S'il fallait limiter je serais pour limiter aux chaines, sans notion de type de données.

~~

> Pour le reste, ça m'a l'air pas mal. Je dois cependant essayer de comprendre encore comme ça s'imbrique avec la notion de "live" (si tu l'as envisagé ou pas)

Non pour le moment l'évaluation est bornée à l'affichage et submit de page; j'imagine possible d'étendre ça pour également faire ça début/fin d'évaluation live mais je rangeais ça pour plus tard.

#8 - 28 avril 2021 18:22 - Frédéric Péters

> Non pour le moment l'évaluation est bornée à l'affichage et submit de page; j'imagine possible d'étendre ça pour également faire ça début/fin d'évaluation live mais je rangeais ça pour plus tard.

En fait j'écrivais ça sans trop réfléchir; en réfléchissant un peu j'arrive à me dire qu'il n'y a rien à faire, que le live se fait sur l'évaluation de la page et aura les champs et que ceux-ci ne changeront pas, mais que j'ai peut-être oublié de justement mettre à disposition ces variables lors de l'évaluation du live. (je vais ajouter un test pour vérifier/corriger).

À un moment j'ai été à me dire qu'il pourrait y avoir d'une manière ou d'une autre recalcul et qu'une valeur changerait et que ça serait utile de l'avoir mais sans arriver à continuer l'idée vers une situation concrète, donc je rangerais vraiment cette part de côté.

#9 - 28 avril 2021 18:29 - Frédéric Péters

> (je vais ajouter un test pour vérifier/corriger).

Voilà j'ai ajouté test_computed_field_usage_in_live_data qui vérifie qu'un commentaire dont le contenu est réévalué par le live a bien la valeur du champ calculé, et ça a marché sans modif.

#10 - 29 avril 2021 03:12 - Thomas Noël

- Fichier computed-data-listing.png ajouté

Frédéric Péters a écrit :

> > la création d'identifiant en doublon me semble être un truc qui va se produire souvent
>
> (...) (Il n'y a de toute façon pas d'interdiction d'identifiant en doublon, de manière générale sur les champs.)

Yep, laissons ainsi, c'est difficile de faire autrement de toute façon. On voit immédiatement sur l'écran le « {{ form_var_xxxx }} » au niveau du champ créée, ça suffit.

> > Idéalement j'aurai préféré que dans ComputedField on demande de défniir un "target_type"
>
> J'imaginais pas exemple qu'un champ pourrait avoir comme contenu le résultat d'un appel web, qui ne rentrerait pas dans ces cases.

Ouaip, ce cas à lui seul annule mon idée. Il va être fréquent et nécessaire. Allez, tant pis, on garde la souplesse avec les risques qui vont avec.

Peut-être à un moment faudra-t-il qu'on distingue dans l'UI les champs d'expression qui ont un allow_complex=True, pour rappeler à l'utilisateur qu'il doit prêter grande attention au type d'objet que son gabarit va générer... mais je n'arrive pas encore à imaginer quelque chose de compréhensible par un boétien.

> > Pour le reste, ça m'a l'air pas mal. Je dois cependant essayer de comprendre encore comme ça s'imbrique avec la notion de "live" (si tu l'as envisagé ou pas)
>
> Voilà j'ai ajouté test_computed_field_usage_in_live_data qui vérifie qu'un commentaire dont le contenu est réévalué par le live a bien la valeur du champ calculé, et ça a marché sans modif.

Génial.

Reste un petit détail, dans la liste de choix du type de champ, le "Computed Data" apparait dans la section titre/sous-titre/commentaire/page (cf copie d'écran jointe). C'est voulu ? J'en ferais plutôt une section à part, comme pour les blocs.

#11 - 29 avril 2021 08:31 - Frédéric Péters

> Reste un petit détail, dans la liste de choix du type de champ, le "Computed Data" apparait dans la section titre/sous-titre/commentaire/page (cf copie d'écran jointe). C'est voulu ? J'en ferais plutôt une section à part, comme pour les blocs.

Oui c'était voulu mais j'avais noté ça dans ma liste de points à reprendre en questions et j'ai oublié; je viens de pousser une branche qui l'isole.

```
@@ -3459,6 +3459,8 @@ def get_field_types():
 def get_field_options(blacklisted_types):
     widgets, non_widgets = [], []
     for klass in field_classes:
+        if klass is ComputedField:
+            continue
         if klass.key in blacklisted_types:
             continue
         if issubclass(klass, WidgetField):
@@ -3466,6 +3468,10 @@ def get_field_options(blacklisted_types):
         else:
             non_widgets.append((klass.key, _(klass.description), klass.key))
     options = widgets + [('', '—', '')] + non_widgets
+
+    # add computed field in its own "section"
+    options.extend([('', '—', ''), (ComputedField.key, _(ComputedField.description), ComputedField.key)])
+
     if get_publisher().has_site_option('fields-blocks') and (
         not blacklisted_types or 'blocks' not in blacklisted_types
     ):
```

#12 - 29 avril 2021 11:05 - Thomas Noël

Suite de mes tests, en pré-visualisation d'un formulaire avec un champ calculé j'ai ce message : « Erreur à l'affichage en prévisualisation du champ. ('NoneType' object is not callable) » parce que « add_to_form = None » pour un ComputedField.

Et donc peut-être faire :

```
--- a/wcs/admin/forms.py
+++ b/wcs/admin/forms.py
@@ -1141,7 +1141,7 @@ class FormDefPage(Directory):
         on_page = 0
         for i, field in enumerate(self.formdef.fields):
             field.id = i
-            if hasattr(field, str('add_to_form')):
+            if getattr(field, str('add_to_form'), None):
                 try:
                     field.add_to_form(form)
                 except Exception as e:
```

#13 - 29 avril 2021 12:25 - Frédéric Péters

if getattr(field, str('add_to_form'), None): et nettoyer ça pour le str() devenu inutile il y a longtemps.

Branche poussée ainsi. Mais ça peut ouvrir la question de se demander si ces champs devraient être d'une manière particulière quand même affiché dans cette "prévisualisation", là j'ai pris l'option de non rien afficher.

#14 - 29 avril 2021 14:12 - Thomas Noël

- Statut changé de Solution proposée à Solution validée

Frédéric Péters a écrit :

> Mais ça peut ouvrir la question de se demander si ces champs devraient être d'une manière particulière quand même affiché dans cette "prévisualisation", là j'ai pris l'option de non rien afficher.

J'y ai pensé aussi mais je propose de laisser les choses ainsi pour l'instant (disons le : d'une façon générale pour moi cette preview ne sert plus à rien... sauf à faire planter la page quand y'a 10 appels webservices sur 5 pages)

Pour moi c'est ok ainsi.

(Grosse évolution, quand même... quel vieux serpent de mer tu as apprivoisé ici...!)

#15 - 30 avril 2021 12:23 - Frédéric Péters

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

```
commit f78d10fd8fce93006f2722eec98e5a8db4594cc4
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Tue Apr 6 08:35:48 2021 +0200

    general: introduce a new "computed data" field (#52110)
```

#16 - 30 avril 2021 19:16 - Frédéric Péters

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

Fichiers

0001-general-introduce-a-new-computed-data-field-52110.patch	35 ko	26 avril 2021	Frédéric Péters
computed-data-listing.png	31,6 ko	29 avril 2021	Thomas Noël