w.c.s. - Development #8265

Blocs de champs

15 septembre 2015 11:22 - Frédéric Péters

Statut:	Fermé	**Début:**	15 septembre 2015
Priorité:	Normal	**Echéance:**	22 juin 2020
Assigné à:	Frédéric Péters	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

-

Demandes liées:

Lié à Publik - Development #19753: Mise en place d'un type de champ "blocs de...	**Fermé**	**29 octobre 2017**
Lié à w.c.s. - Autre #36298: Avoir un widget fichier permettant de joindre pl...	**Fermé**	**20 septembre 2019**
Lié à w.c.s. - Development #43551: ne pas afficher d'erreur quand le bouton "...	**Fermé**	**03 juin 2020**
Lié à w.c.s. - Development #43580: factoriser le code de redirection après su...	**Fermé**	**03 juin 2020**

Révisions associées

Révision 95c65b63 - 30 juin 2020 17:50 - Frédéric Péters

general add support for blocks of fields (#8265)

Historique

#1 - 15 septembre 2015 11:58 - Thomas Noël

Dans mon idée, c'est un type de champ "bloc de champs externes", qui se configure avec:
- une référence à un autre formulaire, dans lequel sont les vrais champs
- un préfixe pour les variables : la variable "bar" dans le formulaire référencé dans le bloc "foo" => form_var_foo_bar

#2 - 29 novembre 2016 17:58 - Thomas Noël

- Priorité changé de Normal à Bas

#3 - 29 octobre 2017 06:38 - Pierre Cros

- Lié à Development #19753: Mise en place d'un type de champ "blocs de champs" ajouté

#4 - 03 janvier 2018 10:29 - Frédéric Péters

Concernant la structure de la db, après avoir vu un formulaire avec >60 pages identiques, avec chacune 20 champs, ce qui commençait à taper sur la limite du nombre de colonnes par table de postgresql, il faudra sans doute prendre en compte ça et que cette notion de "bloc de champ" vienne avec un stockage particulier (soit des tables gérées comme les formdef, soit dans la table du formulaires des colonnes genre json avec l'ensemble du contenu sérialisé).

#5 - 24 septembre 2019 13:07 - Mikaël Ates

- Lié à Autre #36298: Avoir un widget fichier permettant de joindre plusieurs pages ajouté

#6 - 18 novembre 2019 11:00 - Pierre Cros

- Version cible mis à Wishlist

#7 - 11 mai 2020 10:41 - Marie Kuntz

- Priorité changé de Bas à Normal

#8 - 11 mai 2020 11:41 - Marie Kuntz

- Version cible Wishlist supprimé

#9 - 20 mai 2020 09:03 - Frédéric Péters

- Echéance mis à 22 juin 2020

- Assigné à mis à Frédéric Péters

#10 - 03 juin 2020 08:46 - Frédéric Péters

- Lié à Development #43551: ne pas afficher d'erreur quand le bouton "ajouter une ligne" est cliqué ajouté

#11 - 03 juin 2020 15:26 - Frédéric Péters

- Lié à Development #43580: factoriser le code de redirection après suppression/duplication d'un champ ajouté

#12 - 16 juin 2020 16:54 - Frédéric Péters

- Fichier 0001-general-add-support-for-blocks-of-fields-8265.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

wcs/admin/blocks.py sans surprise, le seul point à y noter c'est qu'on n'autorise pas les conditions de champs pour les champs qui sont dans les blocs. C'est une évolution qui sera à envisager mais ne tentons pas trop de choses. (il reste possible de manière globale à un bloc de champ d'être conditionnée). Ici comme pour les données de traitements, utilisation d'uuid pour les champs, pour éviter les clashs de types.

Aussi, on n'y autorise pas les champs de type tableaux, ni d'autres blocs de champs, pas de récursion là.

Modif à wcs/admin/fields.py qui aurait très bien pu aller dans un patch/ticket à part, il s'agit juste par défaut de présenter les tableaux de champs sans préfixe déterminé avec un ..._, plutôt que rien, ça suit ce qui avait été fait pour les champs du profil utilisateur, où on les affiche avec ..._user_var_ devant. C'est mieux d'afficher {{..._plop}} qui indique clairement qu'il manque un bout, et que va être refusé lors de la validation de syntaxe, que {{plop}} qui va passer crème et ne pas marcher.

wcs/admin/forms.py et ajout en haut d'un lien/bouton "blocs de champs", entre catégories et sources de données. Je me dis qu'on peut discuter ici, que ça pourrait être déplacé à la racine, et/ou répété dans les modèles de fiche. (mais on n'y répète pas "sources de données"). À terme ma préférence irait à glisser tout ça dans la page "index" studio, mais c'est mettre un peu trop d'importance sur la fonctionnalité aujourd'hui.

Dans wcs/admin/settings.py l'ajout à l'export global de ces blocs de champs. (il y aussi export/import individuel depuis la page d'un bloc).

wcs/blocks.py pour découvrir un objet tout bête, rien de bien particulier ici, méthodes export/import XML comme on a pour les formulaires et les workflows, etc. Ensuite dans le fichier, BlockSubWidget et BlockWidget, c'est eux qui au final se trouvent affichés, le BlockWidget étant la liste et le BlockSubWidget la "ligne" individuelle de champs.

C'est dans ce fichier qu'on voit la représentation de ces champs, un dictionnaire avec une clé data, contenant une liste, contenant le dictionnaire avec les valeurs (les clés étant les identifiants de champs). C'est une liste même quand la fonctionnalité de pouvoir répéter les champs n'est pas utilisée, pour pouvoir activer ça sans galères.

À côté de la clé data, il y a une clé schema, qui reprend le type des champs, il y a un commentaire pour expliquer le pourquoi; ce qui se passe c'est que champ date est time.struct_time, qu'en mode postgresql avec un champ jsonb ça va être sérialisé sous forme de liste, et désérialisé ça restera une liste, et badaboum. Avoir le schéma permet au stockage et à la lecture de convertir les données.

J'ai à un moment abandonné la colonne jsonb pour mettre à la place un bytea qui aurait contenu la valeur picklée mais j'ai ensuite trouvé ça dommage par rapport aux possibilités de requêtes JSON de postgresql, et donc voilà.

Sur les objets de type bloc aussi, un champ digest_template, pour permettre une représentation textuelle d'une ligne, par exemple on aurait enfant avec prénom nom date de naissance, on pourrait avoir juste "prénom nom" comme représentation, en définissant ici "{{enfant_var_prenom}} {{enfant_var_nom}}". (la première partie étant le slug du bloc lui-même).

wcs/fields.py pour avoir le render_br décrit dans #44139, aussi pour systématiquement attaché le champ au widget, extension systématique des get_view_value pour autoriser un **kwargs, pour le bénéfice du champ fichier, parce qu'on était jusqu'ici assez simplet à juste pouvoir faire ?f=<id> sauf qu'on doit désormais également faire référence à l'éventuel bloc contenant le champ fichier.

Fin du fichier avec BlockField, qui n'a pas grand chose de particulier, et l'ajout à la liste des champs insérables des champs de type "bloc".

Dans wcs/formdef.py la transformation en classmethod de get_field_data (qu'on utilise dans wcs/blocks.py pour récupérer les données des champs), c'est juste la signature qui change, elle n'utilisait de toute façon pas self. Et l'ajout des blocs à la liste des "formdefs", pour que la détection d'utilisation des sources de données passent dessus.

wcs/forms/common.py pour les modifications pour le téléchargement des champs fichier.

wcs/forms/root.py avec la modification à la validation pour mettre pour de faux une méthode GET, pour assurer que Quixote ne vienne pas remplacer des valeurs de champs.

wcs/publisher.py pour la partie import correspondant à l'export global de wcs/admin/settings.py.

wcs/sql.py avec les changements pour le passage de "schéma" lors du jsonb, cf explication plus haut.

wcs/templates/wcs/backoffice/blocks.html totalement inintéressant.

wcs/variables.py pour les accès, seul truc que je vois à noter c'est que sur les champs bloc sans répétition on annonce form_var_block_var_foobar, et avec répétitions form_var_block_0_foobar (et 1, 2, 3), mais les deux fonctionnent dans tous les cas (le *var* prenant juste le premier élément).

#13 - 16 juin 2020 16:58 - Frédéric Péters

Pas dans ce ticket, outre les conditions intra-champs, une présentation "tableau" pour ces blocs de champs, ça se simule avec des grid mais on trouve quand même les libellés des champs répétés. (il y a bien un WidgetListAsTable déjà et ça "fonctionne" mais le rendu force les champs texte, ça demande à revoir davantage).

#14 - 30 juin 2020 01:12 - Thomas Noël

Première relecture, dans un ordre incertain encore, pour la forme et surtout pour comprendre les principes :

La plomberie...

wcs/blocks.py

- BlockDef.get_new_slug : remplacer '%s-%s' par '%s_%s' dans new_slug = '%s-%s' % (base_new_slug, suffix_no)

wcs/sql.py

- ok

wcs/publisher.py

- ok

wcs/variables.py

- ok (mais j'avoue j'ai bien bloqué)

wcs/formdef.py

- ok

L'admin...

wcs/admin/blocks.py

- mini détail, sur le slug, j'aurais bien vu un hint qui explique le readonly quand self.objectdef.is_used()
- blacklisted_* mais je ne dis rien, ma fille dort, t'as de la chance, je lui expliquerais qu'on fera un autre patch.

wcs/admin/fields.py

- yep

wcs/admin/forms.py

- ok (et oui pour, prochainement, mieux se servir de la page studio et éviter les "catégories" et "sources de données" et autres bizarrement situées en haut des formulaires)

wcs/admin/settings.py

- ok

...

tout ça était assez "facile" pour l'instant, je continue de creuser ce mardi. Puis je ferais quelques manip "en vrai" pour voir l'allure générale, mais je n'aurais je pense pas de remarque à faire qu'on ne puisse pas gérer plus tard (genre l'utilisation du menu de studio).

#15 - 30 juin 2020 06:23 - Frédéric Péters

> BlockDef.get_new_slug : remplacer '%s-%s' par '%s_%s' dans new_slug = '%s-%s' % (base_new_slug, suffix_no)

yep, tout à fait, fait.

> mini détail, sur le slug, j'aurais bien vu un hint qui explique le readonly quand self.objectdef.is_used()

yep, voilà.

> blacklisted_* mais je ne dis rien, ma fille dort, t'as de la chance, je lui expliquerais qu'on fera un autre patch.

oui, je ne voulais pas mêler ça au patch.

#16 - 30 juin 2020 12:52 - Thomas Noël

wcs/fields.py

- je n'arrive pas trop à voir comment va être levé un KeyError lors d'un get_field_class_by_type('block:foobar')() si le bloc foobar n'existe pas. C'est nécessaire pour empêcher l'import d'un formulaire (ou d'un form de workflow) qui utiliserait ce type de bloc. Peut-être que ça marche déjà, mais je pense qu'on serait plus à l'aise en levant explicitement un KeyError quelque part dans la classe BlockField (sur la property block ?).
- ajouter un required=False sur add_element_label (parce que add_element_label=add_element_label or _('Add another') dans le widget)

#17 - 30 juin 2020 13:50 - Frédéric Péters

> je n'arrive pas trop à voir comment va être levé un KeyError lors d'un get_field_class_by_type('block:foobar')() si le bloc foobar n'existe pas. (...)

Oui à la base ce code existait déjà, c'est pour la situation où un type de champ inconnu est référencé (par exemple si on ajoute un type de champ "number" en recette, qu'on exporte, qu'on importe ensuite sur la prod), je n'y ai pas touché et il ne détecte de fait pas la tentative lors de l'import d'un formdef d'un champ d'un type de bloc inconnu.

Par contre il manque oui une protection contre l'import d'un formulaire dont le type de champ ne serait pas connu.

Ajoutée; ça donnera "Fichier invalide (Type de champ inconnu [block:XXX]) ce qui est suffisamment explicite je pense.

> ajouter un required=False

C'est required=False par défaut.

#18 - 30 juin 2020 14:32 - Thomas Noël

wcs/forms/common.py

- je ne serais pas contre un petit commentaire minimal genre « # block field » sur les « if '$' in x: fn2, idx, sub = x.split($) »

wcs/forms/root.py

- ok

... et avec ça j'ai fait le tour de la relecture, en gros je n'ai rien eu à dire. Je vais tester "en vrai" maintenant.

#19 - 30 juin 2020 14:36 - Frédéric Péters

> je ne serais pas contre un petit commentaire minimal genre « # block field » sur les « if '$' in x: fn2, idx, sub = x.split($) »

je viens d'ajouter un mini-commentaire. (×2)

#20 - 30 juin 2020 15:30 - Thomas Noël

- Fichier Capture-20200630-145040-662x281.png ajouté

- Fichier Capture-20200630-152715-299x881.png ajouté

À l'usage, quelques idées d'amélioration.

Retour sur wcs/admin/blocks.py :

- dans les settings, préciser par le slug est notamment le préfixe pour le gabarit de résumé, par exemple hint=_('This is notably used as prefix for variable names in Digest Template below.'))
- toujours dans les settings, sur le résumé, préciser que c'est un template :

```
StringWidget, 'digest_template', title=_('Digest'), value=self.objectdef.digest_template, size=50
```

  c'est-à-dire ici juste remplacer 'Digest' par 'Digest Template'.

- et sur la page principale, qui liste les champs d'un bloc, pour un champ avec un identifiant foo c'est {{foo}} qui s'affiche (cf copie d'écran) alors que je pensais qu'on verrait {{..._foo}} ...?

Retour sur wcs/fields.py (BlockField) :

- Je me posais la question du comportement de IntWidget pour max_items : actuellement il peut être vide ou contenir 0. Dans ces cas, on n'a plus de limite du nombre de champs possibles. On devrait éviter cela et imposer un max_items supérieur ou égal à 1.

Concernant wcs/variables. py il pourrait y avoir une difficulté quand, un jour, sur un bloc, on décide de passer de max_items = 1 à max_items > 1 :

- quand un bloc est limité à un seul item, on a des noms form_var_slug_var_foobar
- mais si on passe à un max_items > 1, alors le nom change et devient form_var_slug_var_0_foobar
  Quand max_items > 1 ça serait bien que form_var_slug_var_foobar reste accessible en tant que "premier item" (égal à form_var_slug_var_0_foobar). Et, inversement, quand il n'y a qu'un seul item, que form_var_slug_var_0_foobar existe aussi.

Et tiens, pour savoir combien de champs on été vraiment renseignés, je n'ai trouvé que form_var_zou_raw_data|length. On pourrait trouver plus court ?

Dans l'autre sens, si on avait un bloc avec un max_items = 2 et qu'on le repasse à 1, les demandes qui avaient deux items continuent à montrer les deux (dans le résumé en backoffice et dans form_var_slug). Mais ça pourrait faire l'objet d'un ticket à suivre, je ne sais pas si c'est vraiment génant (ça pourrait être une "feature" de ne pas oublier les anciennes demandes qui avaient plus d'infos).

Dernier truc vu : dans la construction d'un formulaire, quand on déroule la liste des champs possibles, les champs de type bloc sont tout en bas et invisibles, il faut scroller dans la liste pour les faire apparaitre. C'est pas vraiment génant mais ça peut perturber, peut-être, notamment parce qu'on ne voit pas qu'on peut scroller... (seconde copie d'écran jointe).

#21 - 30 juin 2020 15:53 - Frédéric Péters

- Fichier screenshot.png ajouté

> et sur la page principale, qui liste les champs d'un bloc, pour un champ avec un identifiant foo c'est {{foo}} qui s'affiche (cf copie d'écran) alors que je pensais qu'on verrait {{..._foo}} ...?

Ce n'est pas le cas parce que field_var_prefix = '' mais il me semble bien que j'avais mis ça pour ne pas avoir le form_var_ affiché et que j'ai oublié une fois les ..._ repris par défaut d'effacer ça, fait.

> Je me posais la question du comportement de IntWidget pour max_items : actuellement il peut être vide ou contenir 0. Dans ces cas, on n'a plus de limite du nombre de champs possibles. On devrait éviter cela et imposer un max_items supérieur ou égal à 1.

Mais aujourd'hui IntWidget n'est pas surchargé dans wcs/qommon/form.py, ça voudrait dire faire ça pour lui ajouter quelques options. En raccourci je mets if not max_items: max_items = 1.

> Concernant wcs/variables. py il pourrait y avoir une difficulté quand, un jour, sur un bloc, on décide de passer de max_items = 1 à max_items > 1 :

Les différentes formes (soit indexé, soit *var* comme raccourci vers le premier) fonctionnent peu importe le nombre d'éléments, c'est juste ce qui est repris dans l'/inspect qui varie selon la situation, j'allais ajouter un test mais on a bien déjà :

```
    assert variables.get('form_var_block_0_foo') == 'foo'
    assert variables.get('form_var_block_1_foo') == 'foo2'
    assert variables.get('form_var_block_var_foo') == 'foo'  # alias to 1st element
```

> Et tiens, pour savoir combien de champs on été vraiment renseignés, je n'ai trouvé que form_var_zou_raw_data|length. On pourrait trouver plus court ?

En effet, autant pouvoir faire form_var_zou|length, c'est fait via : (et test ajouté)

```
+    def __len__(self):
+        data = self._formdata.data.get(self._field.id)['data']
+        return len(data)
```

> Mais ça pourrait faire l'objet d'un ticket à suivre, je ne sais pas si c'est vraiment génant (ça pourrait être une "feature" de ne pas oublier les anciennes demandes qui avaient plus d'infos).

Oui c'était bien mon idée de ne pas perdre l'info.

> Dernier truc vu : dans la construction d'un formulaire, quand on déroule la liste des champs possibles, les champs de type bloc sont tout en bas et invisibles, il faut scroller dans la liste pour les faire apparaitre. C'est pas vraiment génant mais ça peut perturber, peut-être, notamment parce qu'on ne voit pas qu'on peut scroller... (seconde copie d'écran jointe).

J'ai bien une barre de défilement. (curieux bug de navigateur) (détachons ça de ce ticket mais un jour je reviendrai sur cet ajout de champ, que ça passe par une boite de dialogue, avec les types de champs regroupés, et ça éviter les bugs de <select> de navigateurs).

#22 - 30 juin 2020 17:46 - Thomas Noël

- Statut changé de Solution proposée à Solution validée

Frédéric Péters a écrit :

> En raccourci je mets if not max_items: max_items = 1.

Impec.

> > Concernant wcs/variables. py il pourrait y avoir une difficulté quand, un jour, sur un bloc, on décide de passer de max_items = 1 à max_items > 1 :
>
> Les différentes formes (soit indexé, soit *var* comme raccourci vers le premier) fonctionnent peu importe le nombre d'éléments, c'est juste ce qui est repris dans l'/inspect qui varie selon la situation, j'allais ajouter un test mais on a bien déjà : (...)

Erreur de ma part, je testais un form_var_slug_var_0_foo (avec un _var_ en trop). Tout roule.

> En effet, autant pouvoir faire form_var_zou|length, c'est fait via : (et test ajouté)

Impec.

> > Dernier truc vu : dans la construction d'un formulaire, quand on déroule la liste des champs possibles, les champs de type bloc sont tout en bas et invisibles, il faut scroller dans la liste pour les faire apparaitre. C'est pas vraiment génant mais ça peut perturber, peut-être, notamment parce qu'on ne voit pas qu'on peut scroller... (seconde copie d'écran jointe).
>
> J'ai bien une barre de défilement. (curieux bug de navigateur) (détachons ça de ce ticket mais un jour je reviendrai sur cet ajout de champ, que ça passe par une boite de dialogue, avec les types de champs regroupés, et ça éviter les bugs de <select> de navigateurs).

Ca marche.

Go !

#23 - 30 juin 2020 17:51 - Frédéric Péters

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

```
commit 95c65b6326683e94f94a55f69b17533ce6db5f81
Author: Frédéric Péters <fpeters@entrouvert.com>
Date:   Tue May 19 15:00:02 2020 +0200

    general add support for blocks of fields (#8265)
```

#24 - 30 juin 2020 19:16 - Frédéric Péters

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

Fichiers

0001-general-add-support-for-blocks-of-fields-8265.patch	76,5 ko	16 juin 2020	Frédéric Péters
Capture-20200630-145040-662x281.png	29,2 ko	30 juin 2020	Thomas Noël
Capture-20200630-152715-299x881.png	33 ko	30 juin 2020	Thomas Noël
screenshot.png	29,1 ko	30 juin 2020	Frédéric Péters