w.c.s. - Bug #23718

Page conditonnelle "oubliée"

14 mai 2018 13:55 - Thomas Noël

Statut:	Fermé	**Début:**	14 mai 2018
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Formulaire à deux pages.

Page 1:

- Condition d'affichage : "True"
- Un champ "val1" (donnera un form_var_val1)

Page 2:

- Condition : form_var_val1 [ou bool(vars().get('form_var_val1'))]

Bogue : la page 2 n'est jamais affichée, alors que le formulaire produit contient bien un form_var_val1 non nul...

Notes:

- si on fait "précédent" sur la page de validation, on arrive bien page 2
- si on ajoute une page 3, la page 2 est toujours sautée

Révisions associées

Révision 33aed517 - 14 mai 2018 19:45 - Thomas Noël

evaluates first page condition without breaking the others (#23718)

Historique

#2 - 14 mai 2018 13:58 - Thomas Noël

Notes, encore :

- si on supprime la condition de la page 1, tout marche
- si la condition est "True" alors la page 2 s'affiche bien (ce qui laisserait penser à une absence de la variable form_var_val1 ?)

#3 - 14 mai 2018 13:59 - Thomas Noël

- Fichier condition-finale.wcs ajouté

Mon canevas de test attaché.

#4 - 14 mai 2018 14:08 - Thomas Noël

En déboguant, au moment de l'évaluation de la condition, local_variables contient:

- 'field_val1': 'x'
- 'var_val1': 'x'
  mais
- 'form_field_val1': None
- 'form_var_val1': None

#5 - 14 mai 2018 14:23 - Thomas Noël

Un test qui fait planter :

```
diff --git a/tests/test_form_pages.py b/tests/test_form_pages.py
index 12d28fbe..2d2ae075 100644
```

```
--- a/tests/test_form_pages.py
+++ b/tests/test_form_pages.py
@@ -757,6 +757,22 @@ def test_form_multi_page_condition_on_first_page(pub):
     with pytest.raises(AssertionError):
         resp.form.get('previous')

+def test_form_multi_page_condition_with_form_var(pub):
+    formdef = create_formdef()
+    formdef.fields = [
+            fields.PageField(id='0', label='1st page', type='page',
+                condition={'type': 'python', 'value': 'True'}),
+            fields.StringField(id='1', label='string', varname='val1'),
+            fields.PageField(id='2', label='2nd page', type='page',
+                condition={'type': 'python', 'value': 'vars().get("form_var_val1") == "foo"'}),
+            fields.StringField(id='3', label='string 2')]
+    formdef.store()
+    resp = get_app(pub).get('/test/')
+    formdef.data_class().wipe()
+    resp.forms[0]['f1'] = 'foo'
+    resp = resp.form.submit('submit')
+    assert resp.form['f3']
+
 def test_form_multi_page_condition_no_visible_page(pub):
     formdef = create_formdef()
     formdef.fields = [
```

Résultat :

```
E       AssertionError: No field by the name 'f3' found
```

#6 - 14 mai 2018 14:51 - Thomas Noël

Tout soucis disparait si on ajoute une première page sans condition.

Mon analyse actuelle : tout est lié à la gestion des conditions sur la première page.

Parce que lors de la génération des variables de substitution, on ajoute des ConditionVars ainsi que:

```
        class ConditionVars(object):
            def get_substitution_variables(self):
                return form_live_data
```

```
            def __eq__(self, other):
                # Assume all ConditionVars are equal (as they are because they
                # are created using the same live data); this avoids filling
                # the substitution sources with duplicates and invalidating its
                # cache.
                try:
                    return self.__class__.__name__ == other.__class__.__name__
                except AttributeError:
                    return False
```

Mais si on a une première page avec une condition, on fait ça :

```
        ...
        if self.fields and self.fields[0].type == 'page' and self.fields[0].condition:
            # multipage form with conditions on the first page
            for field in self.fields:
                if field.type != 'page':
                    continue
                if field.is_visible({}, self):    <--- boum
                    break
                ...
```

et donc on génère dans ce cas un premier ConditionVars avec un contenu "{}" dans le formdata, donc tous les form_* à None, qui prendra toujours le dessus sur les ConditionVars des pages suivantes, à cause du "__eq__" qui dit de ne pas ajouter de ConditionVars quand il y en a déjà un.

#7 - 14 mai 2018 16:29 - Thomas Noël

- Fichier 0001-evaluates-first-page-condition-without-breaking-the-.patch ajouté

- Statut changé de Nouveau à En cours

- Patch proposed changé de Non à Oui

Après pas mal de tests... voici le patch qui joue les calculs de conditions en gérant le appels is_visible({}) de façon la plus compatible possible avec

l'existant. Si on est sur la première condition de la première page (et autre appels possibles avec un formdata vide), on va poser dans les variables de l'évaluation les form_* et var_*, qui seront None, mais sans passer par l'utilisation des variables de substitution. Celles-ci ne sont utilisées que lorsqu'un vrai formdata est disponible et envoyé aux conditions (comme actuellement quoi).

J'ai tenté plus petit, mais il y avait toujours des cas non passants dans nos tests («form_var_truc» qui doit échouer quand la variable existe mais est vide, ce genre de chose) et j'ai préféré ne pas les modifier, quand bien même ils étaient un peu «limites».

#8 - 14 mai 2018 16:45 - Frédéric Péters

Pour minimiser le diff je laisserais le ConditionVars à son niveau d'indentation actuel, je ne conditionnerais que le .feed().

J'aurais fait des test sur "is None" (et modifier l'appel .is_visible({}...)

> all will be "None"

Ça fait penser que ce serait un None transformé par erreur en chaine de caractères :/ Peut-être "(they will all have been set to None)". Et peut-être inclure le commentaire à l'intérieur de la condition ?

Et plutôt, reprise total, proposition pour davantage d'explications :

```
  # ConditionsVars is not set when evaluating first page but we need to have form_var_*
  # variables already; add them from form_live_data (where all variables will have been
  # set to None).
```

#9 - 14 mai 2018 20:03 - Thomas Noël

- Fichier 0001-evaluates-first-page-condition-without-breaking-the-.patch ajouté

Frédéric Péters a écrit :

> Pour minimiser le diff je laisserais le ConditionVars à son niveau d'indentation actuel, je ne conditionnerais que le .feed().

Fait. J'avais bougé plus de trucs pour rendre la fonction plus lisible à mes yeux, mais ça ira très bien comme ça.

> J'aurais fait des test sur "is None" (et modifier l'appel .is_visible({}...)

Fait. Au passage ça oblige à nettoyer ce moment bizarre, et ça très bien :

```
     def is_visible(self, dict, formdef):
-        if dict is None:
-            return True
         try:
             ...
```

> Et plutôt, reprise totale, proposition pour davantage d'explications :

Yep yep. Thanks a lot.

#10 - 15 mai 2018 10:53 - Frédéric Péters

Go.

#11 - 15 mai 2018 11:13 - Thomas Noël

- Statut changé de En cours à Résolu (à déployer)

```
commit 33aed517eed0989bddeffee4b854f3044ead0886
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Mon May 14 15:32:09 2018 +0200

    evaluates first page condition without breaking the others (#23718)
```

#14 - 23 décembre 2018 14:40 - Frédéric Péters

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

Fichiers

condition-finale.wcs	2,05 ko	14 mai 2018	Thomas Noël
0001-evaluates-first-page-condition-without-breaking-the-.patch	4,94 ko	14 mai 2018	Thomas Noël

0001-evaluates-first-page-condition-without-breaking-the-.patch	4,55 ko	14 mai 2018	Thomas Noël