w.c.s. - Development #29406

assouplir la comparaison de form_var_xxx de type date ou datetime

02 janvier 2019 16:35 - Thomas Noël

Statut:	Fermé	**Début:**	02 janvier 2019
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

Via des SimpleLazyObject proposés par Django, on peut surcharger eq/ne/gt/lt/etc. et faire en sorte que "now" et "today" acceptent d'être comparer à date ou datetime, indifféremment.

Révisions associées

Révision e6969acb - 28 janvier 2019 15:09 - Thomas Noël

variables: add flexible comparison on lazy dates (#29406)

Révision e2260c5a - 28 janvier 2019 15:09 - Thomas Noël

misc: add lazydateobject, use it for today and now global variables (#29406)

Révision 87cae6fb - 28 janvier 2019 15:09 - Thomas Noël

make date and datetime templatetags render lazydate objects (#29406)

Révision 4daf8310 - 28 janvier 2019 15:19 - Thomas Noël

evalutils: handle lazy dates in make_date and make_datetime (#29406)

Historique

#1 - 03 janvier 2019 00:04 - Thomas Noël

- Fichier 0001-draft-add-flexible-comparison-on-lazy-dates.patch ajouté

- Statut changé de Nouveau à Information nécessaire

- Patch proposed changé de Non à Oui

Au détour de travaux sur le sujet, je tente de rendre les dates lazy (LazyFieldVarDate) plus souples quant aux comparaisons, notamment pour ne plus être embêté par les différences entre date et datetime.

Le patch ci-joint montre une limite que j'ai du mal à franchir : je peux faire un opérateur __eq__ pour lazyvar foo, mais ça ne marche pas pour foo lazyvar si foo a déjà un opérateur __eq__... comme c'est le cas pour datetime.date, et zut. Et donc, le test dans le patch joint échoue :

```
        # flexible date comparison
        for date in ('2018-07-31', '31/07/2018',
                     datetime.date(2018, 7, 31),
                     datetime.datetime(2018, 7, 31, 0, 0),
                     time.strptime('2018-07-31', '%Y-%m-%d')):
            assert lazy_formdata.var.datefield == date
>           assert date == lazy_formdata.var.datefield
E           assert datetime.date(2018, 7, 31) == <wcs.variables.LazyFieldVarDate object at 0x7f7470850c90>
```

Je ne suis pas sûr qu'on puisse sortir proprement de ce soucis ; si ça se confirme je préférerai revenir à l'idée de "pour comparer des dates il faut toujours utiliser |date ou |datetime" au lieu de tenter d'expliquer dans quel ordre il faut faire les comparaisons (ça sera inexplicable).

Mais peut-être que je rate quelque chose dans le LazyFieldVarDate::__eq__ ou ailleurs.

#2 - 03 janvier 2019 00:42 - Thomas Noël

- Fichier 0001-draft-add-flexible-comparison-on-lazy-dates.patch ajouté

- Statut changé de Information nécessaire à En cours

Et donc, curieux et un peu insomniaque, dans les sources de Python 2 sur la comparaison des date :

```
 ...
    else if (PyObject_HasAttrString(other, "timetuple")) {
        /* A hook for other kinds of date objects. */
        Py_INCREF(Py_NotImplemented);
        return Py_NotImplemented;
    }
 ...
```

et effectivement, si on ajoute un timetuple au LazyFieldVarDate, ça lève un NotImplemented sur le __eq__ des datetime, et hop, ça force notre __eq__ ! Le patch joint ajoute juste :

```
+    def timetuple(self):
+        return self.get_raw()
```

Mais j'ai maintenant un autre soucis amusant sur la comparaison avec un timetuple, qui redevient un simple tuple au moment d'être envoyé dans le __eq__ (et donc parse_datetime se plante) :

```
        # flexible date comparison
        for date in ('2018-07-31', '31/07/2018',
                     datetime.date(2018, 7, 31),
                     datetime.datetime(2018, 7, 31, 0, 0),
                     time.strptime('2018-07-31', '%Y-%m-%d')):
            assert lazy_formdata.var.datefield == date
>           assert date == lazy_formdata.var.datefield

tests/test_formdata.py:639: 
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _
wcs/variables.py:447: in __eq__
    return parse_datetime(self.get_raw()) == parse_datetime(other)
wcs/qommon/templatetags/qommon.py:65: in parse_datetime
    return evalutils.make_datetime(datetime_string)
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _

datetime_var = (2018, 7, 31, 0, 0, 0, ...)

    def make_datetime(datetime_var):
        '''Extract a date from a datetime, a date, a struct_time or a string'''
        if isinstance(datetime_var, datetime.datetime):
            return datetime_var
        if isinstance(datetime_var, datetime.date):
            return datetime.datetime(year=datetime_var.year, month=datetime_var.month,
                                     day=datetime_var.day)
        if isinstance(datetime_var, time.struct_time):
            return datetime.datetime(*datetime_var[:6])
        if hasattr(datetime_var, 'decode'):
            return get_as_datetime(datetime_var)
>       raise ValueError('invalid datetime value: %s' % datetime_var)
E       TypeError: not all arguments converted during string formatting

wcs/qommon/evalutils.py:57: TypeError
```

Étonnant, non ?

Je suis tenté par patcher make_datetime avec if isinstance(datetime_var, (time.struct_time, tuple)): mais c'est pas très joli (et j'aimerai comprendre, aussi).

#3 - 03 janvier 2019 00:44 - Thomas Noël

Tout passe avec :

```
diff --git a/wcs/qommon/evalutils.py b/wcs/qommon/evalutils.py
index 619a19d4..235fa8ce 100644
--- a/wcs/qommon/evalutils.py
+++ b/wcs/qommon/evalutils.py
@@ -35,7 +35,7 @@ def make_date(date_var):
         return date_var.date()
     if isinstance(date_var, datetime.date):
         return date_var
-    if isinstance(date_var, time.struct_time):
+    if isinstance(date_var, (time.struct_time, tuple)):
         return datetime.date(*date_var[:3])
     try:
         return get_as_datetime(str(date_var)).date()
```

```
@@ -50,7 +50,7 @@ def make_datetime(datetime_var):
     if isinstance(datetime_var, datetime.date):
         return datetime.datetime(year=datetime_var.year, month=datetime_var.month,
                                  day=datetime_var.day)
-    if isinstance(datetime_var, time.struct_time):
+    if isinstance(datetime_var, (time.struct_time, tuple)):
         return datetime.datetime(*datetime_var[:6])
     if hasattr(datetime_var, 'decode'):
         return get_as_datetime(datetime_var)
```

mais ça m'interpelle quand même.

#4 - 03 janvier 2019 09:39 - Frédéric Péters

(et j'ai regardé Python 3.6 et ça semble similaire)

Je vérifierais quand même la bonne longueur de tuple.

#5 - 09 janvier 2019 17:25 - Thomas Noël

- Fichier 0001-variables-add-flexible-comparison-on-lazy-dates-2940.patch ajouté

- Sujet changé de permettre à now et today d'être comparer à des date ou des datetime à assouplir la comparaison de form_var_xxx de type date ou datetime

- Statut changé de En cours à Solution proposée

Voici mon idée pour que les form_var_date disposent d'opérateurs de comparaison leur permettant d'être comparer à des date, des datetime, des choses qui ressemblent à des dates (tant que c'est interprétable par parse_datetime)

Pour que ça ait vraiment un usage dans de vraies expressions et conditions, il faudra compléter l'affaire, dans d'autres tickets, avec :

- la création d'un LazyDate du même genre, mais pas lié à un formdata, à utiliser en résultat de |date, de now, de today, etc...
- et l'usage de ce LazyDate aussi en résultat des |add_days et |add_hours à venir

#6 - 14 janvier 2019 16:45 - Frédéric Péters

- Statut changé de Solution proposée à Solution validée

Hop go.

#7 - 17 janvier 2019 16:40 - Thomas Noël

- Fichier 0001-variables-add-flexible-comparison-on-lazy-dates-2940.patch ajouté

- Fichier 0002-misc-add-lazydateobject-use-it-for-today-and-now-glo.patch ajouté

- Statut changé de Solution validée à Solution proposée

En fait je n'étais pas très satisfait par ce patch. En voici une nouvelle forme, qui permet une extension assez simple ensuite vers la lazyfication de "now" et "today" (patch 0002).

C'est poussé dans http://git.entrouvert.org/wcs.git?h=wip/29406-lazy-date-system

#8 - 18 janvier 2019 09:17 - Frédéric Péters

Dans 0001, pour aérer un peu, j'ajouterais une ligne blanche au-dessus de :

```
    for date in ('2018-08-31', '2018-07-31 01:00', '2018-07-31 01:00:00',
```

Pour 0002, on n'a actuellement pas de test assurant le rendu {{today}}, ou {{now}}, ça me rassurerait.

#9 - 18 janvier 2019 14:04 - Thomas Noël

Frédéric Péters a écrit :

> Pour 0002, on n'a actuellement pas de test assurant le rendu {{today}}, ou {{now}}, ça me rassurerait.

Yep ; j'ai isolé ça dans un #29887 à pousser en amont de ce travail.

#10 - 18 janvier 2019 14:22 - Frédéric Péters

- Statut changé de Solution proposée à Solution validée

Ainsi donc ok.

#11 - 18 janvier 2019 14:23 - Thomas Noël

- Fichier 0004-make-date-and-datetime-templatetags-render-lazydate-.patch ajouté

- Fichier 0003-misc-add-lazydateobject-use-it-for-today-and-now-glo.patch ajouté

- Fichier 0002-variables-add-flexible-comparison-on-lazy-dates-2940.patch ajouté

- Statut changé de Solution validée à Solution proposée

Donc, d'abord valider #29887 pour se couvrir un peu (tests supplémentaires).

Puis ces trois patches à lire et valider :

- 0002-variables-add-flexible-comparison-on-lazy-dates-2940.patch : lazy-fication des form_var_date
  - ici petit détail, dans le strftime on doit de-lazyfier avec un copy avant d'envoyer l'objet à datetime_safe.strftime de Django, qui joue avec type() (et les objects Lazy ne savent pas mentir sur leur type)
- 0003-misc-add-lazydateobject-use-it-for-today-and-now-glo.patch : création d'une classe LazyDateObject et utilisation par "now" et "today"
- 0004-make-date-and-datetime-templatetags-render-lazydate-.patch : utilisation de cette lazyfication dans |date et |datetime

La suite c'est l'arrivée des calculs autour des dates (add_days & co) dans #29337

#12 - 21 janvier 2019 22:48 - Thomas Noël

- Fichier 0001-misc-emit-static-non-lazy-today-and-now-variables-29.patch ajouté

Benjamin avait vu juste, l'affaire met un gros bazar dans les calculs en Python qui utiliserait today et now, parce que ce sont des variables d'un nouveau type Lazy. Et par exemple une condition du genre "utils.age_in_days(today) == 0" (un peu idiot mais bon) crashe sur un « TypeError: unsupported operand type(s) for -: 'datetime.date' and 'LazyDateObject' »

Pour corriger ça, je proposerai d'ajouter dans QommonPublisher une méthode pour re-proposer les vrais today et now en mode non-lazy :

```
+++ b/wcs/qommon/publisher.py
@@ -1061,6 +1061,12 @@ class QommonPublisher(Publisher, object):
         d['manager_homepage_title'] = d.get('portal_agent_title')
         return d

+    def get_static_substitution_variables(self):
+        return {
+            'today': datetime.date.today(),
+            'now': datetime.datetime.now(),
+        }
+
```

les tests sont ok, mais ... mon sentiment : tout ça devient assez illisible. Bien que pleine de tests, cette branche me semble un nid à bogues. Mais je suis de nature inquiète...?

Ci-joint le patch qui serait à inclure dans la branche, à merger dans « misc: add lazydateobject, use it for today and now global variables (#29406) ». Je l'ai ajouté dans la branche pour voir si les tests passent via jenkins2.

#13 - 21 janvier 2019 22:58 - Frédéric Péters

> Pour corriger ça, je proposerai d'ajouter dans QommonPublisher une méthode pour re-proposer les vrais today et now en mode non-lazy :

Bof, pas envie de multiplier les différences entre lazy/pas lazy, l'objectif initial était de dégager totalement le mode non-lazy, c'est par conservatisme qu'il a été laissé, et je serais pour un flag permettant aux nouvelles installations de fonctionner totalement avec lazy.

Sur l'exemple soulevé, utils.age_in_days(today), et toutes les fonctions dans evalutils.py, ça doit passer par make_date/make_datetime, il y aurait plutôt là à gérer un type supplémentaire.

#14 - 21 janvier 2019 23:08 - Thomas Noël

Frédéric Péters a écrit :

> (...) ça doit passer par make_date/make_datetime, il y aurait plutôt là à gérer un type supplémentaire.

Tu veux dire, faire en sorte que ça sache gèrer les LazyDate ?

#15 - 21 janvier 2019 23:17 - Frédéric Péters

> Tu veux dire, faire en sorte que ça sache gèrer les LazyDate ?

Oui.

#16 - 21 janvier 2019 23:21 - Thomas Noël

C'était assez facile via «l'astuce» du date/datetime.replace() pour dé-lazyfier :

```
+++ b/wcs/qommon/evalutils.py
@@ -34,7 +34,7 @@ def make_date(date_var):
     if isinstance(date_var, datetime.datetime):
         return date_var.date()
     if isinstance(date_var, datetime.date):
-        return date_var
+        return date_var.replace()  # use replace to un-lazy
     if isinstance(date_var, time.struct_time) or (
             isinstance(date_var, tuple) and len(date_var) == 9):
         return datetime.date(*date_var[:3])
@@ -47,7 +47,7 @@ def make_date(date_var):
 def make_datetime(datetime_var):
     '''Extract a date from a datetime, a date, a struct_time or a string'''
     if isinstance(datetime_var, datetime.datetime):
-        return datetime_var
+        return datetime_var.replace()  # use replace to un-lazy
     if isinstance(datetime_var, datetime.date):
         return datetime.datetime(year=datetime_var.year, month=datetime_var.month,
                                  day=datetime_var.day)
```

Branche mise à jour pour voir les tests réussir, on y croit.

#17 - 22 janvier 2019 00:20 - Thomas Noël

Voilà, branche à jour avec les 5 patches, finalement j'ai laissé le dernier "à part", il a sa petite logique à lui :
http://git.entrouvert.org/wcs.git?h=wip/29406-lazy-date-system

#18 - 25 janvier 2019 17:31 - Frédéric Péters

- Statut changé de Solution proposée à Solution validée

#19 - 28 janvier 2019 22:50 - Thomas Noël

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

```
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Mon Jan 21 22:44:35 2019 +0100

    evalutils: handle lazy dates in make_date and make_datetime (#29406)

commit 87cae6fbbcb44d5844b51304e021b7ce98ce6444
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Thu Jan 17 17:19:42 2019 +0100

    make date and datetime templatetags render lazydate objects (#29406)

commit e2260c5ac07c1e4fb30484c81787068a59bd6374
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Thu Jan 17 16:34:33 2019 +0100

    misc: add lazydateobject, use it for today and now global variables (#29406)

commit e6969acb1c1fbfb35aa990d26216c5865fa8d312
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Wed Jan 2 23:54:03 2019 +0100

    variables: add flexible comparison on lazy dates (#29406)
```

#20 - 28 janvier 2019 23:15 - Frédéric Péters

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

Fichiers

0001-draft-add-flexible-comparison-on-lazy-dates.patch	2,43 ko	02 janvier 2019	Thomas Noël
0001-draft-add-flexible-comparison-on-lazy-dates.patch	2,6 ko	02 janvier 2019	Thomas Noël
0001-variables-add-flexible-comparison-on-lazy-dates-2940.patch	12,7 ko	09 janvier 2019	Thomas Noël
0001-variables-add-flexible-comparison-on-lazy-dates-2940.patch	13,2 ko	17 janvier 2019	Thomas Noël
0002-misc-add-lazydateobject-use-it-for-today-and-now-glo.patch	3,98 ko	17 janvier 2019	Thomas Noël
0004-make-date-and-datetime-templatetags-render-lazydate-.patch	6,36 ko	18 janvier 2019	Thomas Noël
0003-misc-add-lazydateobject-use-it-for-today-and-now-glo.patch	3,99 ko	18 janvier 2019	Thomas Noël
0002-variables-add-flexible-comparison-on-lazy-dates-2940.patch	13,8 ko	18 janvier 2019	Thomas Noël
0001-misc-emit-static-non-lazy-today-and-now-variables-29.patch	3,09 ko	21 janvier 2019	Thomas Noël