w.c.s. - Development #23305

sql: permettre à select() de fonctionner comme un itérateur

20 avril 2018 12:48 - Thomas Noël

Statut:	Fermé	**Début:**	20 avril 2018
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	

Description

avec un explicite iterator=True à envoyer dans les arguments, posé à False par défaut.

Demandes liées:

Lié à w.c.s. - Development #22383: migrate : séparer les deux étapes (migrati...	**Fermé**	**08 mars 2018**

Révisions associées

Révision f7242954 - 20 avril 2018 13:59 - Thomas Noël

sql: make select a possible iterator (#23305)

Historique

#1 - 20 avril 2018 12:49 - Thomas Noël

- Lié à Development #22383: migrate : séparer les deux étapes (migrations / réindexations) ajouté

#2 - 20 avril 2018 13:08 - Thomas Noël

- Fichier 0001-sql-make-select-a-possible-iterator-23305.patch ajouté

- Patch proposed changé de Non à Oui

#3 - 20 avril 2018 13:19 - Thomas Noël

- Fichier 0001-sql-make-select-a-possible-iterator-23305.patch ajouté

- Statut changé de Nouveau à En cours

Amendement pour éventuellement préparer un usage plus général, j'ajoute un paramétre iterator dans le select de qommon.storage, sans usage (pour l'instant ?)

Interdiff :

```
--- a/wcs/qommon/storage.py
+++ b/wcs/qommon/storage.py
@@ -244,7 +244,8 @@ class StorableObject(object):

     @classmethod
     def select(cls, clause=None, order_by=None, ignore_errors=False,
-            ignore_migration=False, limit=None, offset=None):
+            ignore_migration=False, limit=None, offset=None, iterator=False):
+        # iterator: only for compatibility with sql select()
         keys = cls.keys()
         objects = (cls.get(k, ignore_errors=ignore_errors,
             ignore_migration=ignore_migration) for k in keys)
```

#4 - 20 avril 2018 13:30 - Frédéric Péters

Dans les variations for iterator in (False, True): j'ajouterais une seconde boucle qui serait for func_clause in (lambda x: True, None):.

#5 - 20 avril 2018 13:43 - Thomas Noël

- Fichier 0001-sql-make-select-a-possible-iterator-23305.patch ajouté

Yep. Version avec ça en plus :

```
--- a/tests/test_sql.py
+++ b/tests/test_sql.py
@@ -711,6 +711,7 @@ def test_sql_table_select_iterator():
         # TypeError: object of type 'generator' has no len()

     assert len(list(data_class.select(iterator=True))) == 50
+    assert len(list(data_class.select(lambda x: True, iterator=True))) == 50
     assert len(list(data_class.select(lambda x: x.id < 26, iterator=True))) == 25
     assert len(list(data_class.select([st.Less('id', 26)], iterator=True))) == 25
     assert len(list(data_class.select([st.Less('id', 25), st.GreaterOrEqual('id', 10)],
@@ -759,12 +760,13 @@ def test_select_limit_offset():

     assert len(data_class.select()) == 50
     for iterator in (False, True):
-        assert [x.id for x in data_class.select(order_by='id', iterator=iterator)] == range(1, 51)
-        assert [x.id for x in data_class.select(order_by='id', limit=10, iterator=iterator)] == range(1, 11)
-        assert [x.id for x in data_class.select(order_by='id', limit=10, offset=10, iterator=iterator)] == ra
nge(11, 21)
-        assert [x.id for x in data_class.select(order_by='id', limit=20, offset=20, iterator=iterator)] == ra
nge(21, 41)
-        assert [x.id for x in data_class.select(order_by='id', offset=10, iterator=iterator)] == range(11, 51
)
-        assert len([x.id for x in data_class.select(lambda x: x.id > 10, limit=10, iterator=iterator)]) == 10
+        for func_clause in (lambda x: True, None):
+            assert [x.id for x in data_class.select(func_clause, order_by='id', iterator=iterator)] == range(
1, 51)
+            assert [x.id for x in data_class.select(func_clause, order_by='id', limit=10, iterator=iterator)]
 == range(1, 11)
+            assert [x.id for x in data_class.select(func_clause, order_by='id', limit=10, offset=10, iterator
=iterator)] == range(11, 21)
+            assert [x.id for x in data_class.select(func_clause, order_by='id', limit=20, offset=20, iterator
=iterator)] == range(21, 41)
+            assert [x.id for x in data_class.select(func_clause, order_by='id', offset=10, iterator=iterator)
] == range(11, 51)
+            assert len([x.id for x in data_class.select(lambda x: x.id > 10, limit=10, iterator=iterator)]) =
= 10
```

#6 - 20 avril 2018 13:50 - Frédéric Péters

Le dernier pourrait sortir de la boucle :

```
+            assert len([x.id for x in data_class.select(lambda x: x.id > 10, limit=10, iterator=iterator)]) =
= 10
```

Et avec ça, go.

#7 - 20 avril 2018 14:01 - Thomas Noël

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

Hop, une petite dés-indentation et c'est fait ainsi ; merci.

```
commit f724295485fa1ebcbbd05d9b272eb9ef9f94355e
Author: Thomas NOEL <tnoel@entrouvert.com>
Date:   Fri Apr 20 12:49:58 2018 +0200

    sql: make select a possible iterator (#23305)
```

#8 - 23 décembre 2018 14:40 - Frédéric Péters

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

Fichiers

0001-sql-make-select-a-possible-iterator-23305.patch	6,12 ko	20 avril 2018	Thomas Noël
0001-sql-make-select-a-possible-iterator-23305.patch	6,78 ko	20 avril 2018	Thomas Noël
0001-sql-make-select-a-possible-iterator-23305.patch	6,99 ko	20 avril 2018	Thomas Noël