w.c.s. - Development #57066

faire fonctionner filter_by et filter_value sur des listes de dictionnaires

18 septembre 2021 02:48 - Thomas Noël

Statut:	Nouveau	**Début:**	18 septembre 2021
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	Non

Description

(ticket pour discuter de l'idée et de sa faisabilité)

On a souvent dans nos objets des listes de dictionnaires, et aujourd'hui on aime faire des calculs dans w.c.s. Un truc difficile à faire en Django (voire impossible sur une condition) c'est de filtrer les éléments selon un certain critère. Typiquement on aimerait pouvoir faire un filter_by+filter_value avec la même syntaxe que dans un filtre de requête.

Il me semble que c'est techniquement possible en modifiant un peu filter_by pour qu'il repère quand il reçoit une liste de dicos, créé alors une instance d'une classe « FilterableList(list) » qui retiendra le filter_by et disposera d'une méthode apply_filter_value.

Reste à savoir si mon idée est pertinente. Pour moi oui parce que ça fait un truc en moins qui nécessite du Python (et j'ai déjà un cas d'usage réel).

PS : ces listes de dictionnaires arrivent souvent en réponse d'appel webservices, donc on peut aussi demander à ce que l'API distance gère le filtrage... mais quand le filtre est anecdotique, avoir le filter_by/value ça serait bien pratique et plutôt facile à expliquer/documenter)

Historique

#2 - 18 septembre 2021 09:34 - Frédéric Péters

J'ai déjà du plusieurs fois expliquer que |filter_by et cie étaient des filtres sur des requêtes, que ça ne s'appliquait pas à "n'importe quoi". Donc j'imagine que ça serait utilisé. (j'essaierai de retrouver ces moments passés).

#6 - 18 septembre 2021 09:51 - Frédéric Péters

Dans ce que je retrouve, #54530, il y avait souhait de |filter_ sur les sources de données, avec cet exemple :

```
data_source.imio_townstreet_issuess|filter_by:"categories"|filter_value:"textiles_banks"
```

Et même affaire #54048,

```
{{ data_source.passeport_cni_constitution_creneaux_du_type_1_personne|filter_by:"disabled"|filter_value:"false
"|count }}
```

où j'écrivais :

> Plutôt négatif sur l'idée d'étendre les filtres de requêtes à des structures qui ne sont pas des requêtes, ça sera nécessairement partiel, avec des différences de comportement bizarre.

Pour aller contre cette objection, il y aurait sans doute à préciser dès ce ticket qu'il y a aussi à gérer |order_by et |exclude_value.

#7 - 19 septembre 2021 01:40 - Thomas Noël

Je pensais à une approche très basique de ce genre :

```
+class ListQuerySet(list):
+    filter_by = None
+
+    def apply_filter_value(self, value):
+        for item in self:
+            if item.get(self.filter_by) == value:
+                yield item
+
+    def apply_exclude_value(self, value):
```

```
+         for item in self:
+             if item.get(self.filter_by) != value:
+                 yield item
+
+
 @register.filter
 def filter_by(queryset, attribute):
-    return queryset.filter_by(unlazy(attribute))
+    if hasattr(queryset, 'filter_by'):
+        return queryset.filter_by(unlazy(attribute))
+    queryset = ListQuerySet(queryset)
+    queryset.filter_by = unlazy(attribute)
+    return queryset
```

(et un truc un peu du même genre pour le order_by)

Est-ce que je rate quelque chose, une situation qui ferait que ça ne marcherait pas ?

#8 - 19 septembre 2021 09:21 - Frédéric Péters

Je ne sais pas ce que ça donne avec data_source.whatever en entrée, il manque order_by, et cette implémentation dédoublée (ne pas utiliser les classes Criteria) double le taf sur un changement comme #56089 (adaptation de filter_by aux valeurs de type liste), double le taf mais surtout fait qu'on passera à côté et amène le "ça sera nécessairement partiel, avec des différences de comportement bizarre".