Authentic 2 - Development #36365

API de création d'un utilisateur, prendre pour un attribut vide un champ date qui serait une chaine vide

23 septembre 2019 15:22 - Frédéric Péters

Statut:	Fermé	**Début:**	23 septembre 2019
Priorité:	Normal	**Echéance:**	
Assigné à:	Nicolas Roche	**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Oui	**Planning:**	Non

Description

Parce qu'envoyé à partir d'un gabarit via w.c.s. on ne peut pas envoyer null.

Révisions associées

Révision 05340b11 - 03 octobre 2019 16:11 - Nicolas Roche

api: extend DRF date field to accept empty string (#36365)

Historique

#2 - 23 septembre 2019 17:47 - Nicolas Roche

- Assigné à mis à Nicolas Roche

Voici 2 tests qui valideront que les champs dates réagissent comme le champ text.
(J'ai noté là où le comportement diffère actuellement.)

```
def test_api_users_required_date_attributes(settings, app, admin, simple_user):
    Attribute.objects.create(kind='string', name='prefered_color', label='prefered color', required=True)
    Attribute.objects.create(kind='date', name='date', label='date', required=True)
    Attribute.objects.create(kind='birthdate', name='birthdate', label='birthdate', required=True)

    User = get_user_model()
    payload = {
        'username': simple_user.username,
        'id': simple_user.id,
        'email': 'john.doe@nowhere.null',
        'first_name': 'Johnny',
        'last_name': 'Doe',
    }
    headers = basic_authorization_header(admin)
    resp = app.put_json('/api/users/{}/'.format(simple_user.uuid),
            params=payload, headers=headers, status=400)
    assert resp.json['result'] == 0
    assert resp.json['errors']['prefered_color'] == ['This field is required.']
    assert resp.json['errors']['date'] == ['This field is required.']
    assert resp.json['errors']['birthdate'] == ['This field is required.']

    payload['prefered_color'] = 'blue'
    payload['date'] = '1515-1-15'
    payload['birthdate'] = '1900-2-22'
    resp = app.put_json('/api/users/{}/'.format(simple_user.uuid),
            params=payload, headers=headers, status=200)
    resp = app.get('/api/users/{}/'.format(simple_user.uuid),
            headers=headers, status=200)
    assert resp.json['prefered_color'] == 'blue'
    assert resp.json['date'] == '1515-01-15'
    assert resp.json['birthdate'] == '1900-02-22'

    payload['prefered_color'] = ''
    payload['date'] = ''
    payload['birthdate'] = ''
    resp = app.put_json('/api/users/{}/'.format(simple_user.uuid),
            params=payload, headers=headers, status=400)
    assert resp.json['result'] == 0
    assert resp.json['errors']['prefered_color'] == ['This field may not be blank.']
```

```
    assert resp.json['errors']['date'] == ['This field may not be blank.']           # <- ici (Date has wrong
format...)
    assert resp.json['errors']['birthdate'] == ['This field may not be blank.']      # <- ici (idem)

def test_api_users_optional_date_attributes(settings, app, admin, simple_user):
    Attribute.objects.create(kind='date', name='date', label='date', required=False)
    Attribute.objects.create(kind='birthdate', name='birthdate', label='birthdate', required=False)

    User = get_user_model()
    payload = {
        'username': simple_user.username,
        'id': simple_user.id,
        'email': 'john.doe@nowhere.null',
        'first_name': 'Johnny',
        'last_name': 'Doe',
    }
    headers = basic_authorization_header(admin)
    resp = app.put_json('/api/users/{}/'.format(simple_user.uuid),
            params=payload, headers=headers, status=200)
    resp = app.get('/api/users/{}/'.format(simple_user.uuid),
            headers=headers, status=200)
    assert 'prefered_color' not in resp.json.keys()
    assert 'date' not in resp.json.keys()                # <- ici
    assert 'birthddate' not in resp.json.keys()          # <- ici

    payload['prefered_color'] = ''
    payload['date'] = ''
    payload['birthdate'] = ''
    resp = app.put_json('/api/users/{}/'.format(simple_user.uuid),
            params=payload, headers=headers, status=200)             # <- ici (Date has wrong format...)
    resp = app.get('/api/users/{}/'.format(simple_user.uuid),
            headers=headers, status=200)
    assert 'prefered_color' not in resp.json.keys()
    assert 'date' not in resp.json.keys()
    assert 'birthddate' not in resp.json.keys()
```

#3 - 24 septembre 2019 20:04 - Nicolas Roche

- Fichier 0001-api-extend-DRF-date-field-to-accept-empty-string-363.patch ajouté

- Statut changé de Nouveau à Solution proposée

- Patch proposed changé de Non à Oui

Désolé, le second test ci-dessus est faux : j'ai oublié d'ajouter l'attribut témoin "prefered_color".

En fait, le comportement observé correspond à celui défini par Django-Rest-Framework : rien n'est prévu pour les dates nulles.
https://www.django-rest-framework.org/api-guide/fields/#datefield

Le patch proposé ajoute un champ "allow_blank" au type DateField de Django-Rest-Frameword, afin d'harmoniser le comportement des champs d'A2. 'date' et 'birthdate' se comportent à présent (sur ce point) comme le champs 'string'.

Pour accepter la chaîne vide, je suis intervenu avant la fonction virtuelle 'run_validation' (qui appelle la méthode 'to_internal_value' vérifiant le format ISO_8601).

D'autres options sont envisageables : surcharger 'to_internal_value' ou encore dériver de CharField (ici j'ai préféré copier/coller le minimum requis depuis *rest_framework/fields.py::CharField*).

#4 - 25 septembre 2019 08:44 - Benjamin Dauvergne

Il me semble que ce bout de code n'est pas nécessaire car il sera fait par super().run_validation() :

```
        # Test for the empty string here so that it does not get validated,
        # and so that subclasses do not need to handle it explicitly
        # inside the `to_internal_value()` method.
        (is_empty_value, data) = self.validate_empty_values(data)
        if is_empty_value:
            return data
```

non ? En tout cas dans CharField ça n'est pas répété.

#5 - 25 septembre 2019 11:55 - Nicolas Roche

- Fichier 0001-api-extend-DRF-date-field-to-accept-empty-string-363.patch ajouté

Oui, bien vu (j'ai du le rajouter à un moment où ça ne passait pas, alors que mon patch n'était pas abouti).

J'avais un doute sur le fait qu'il me manque le pendant de la correction côté CSV.
Notamment, comme pour #35647.

api_views.py:

```
class BaseUserSerializer(serializers.ModelSerializer):
...
    def __init__(self, *args, **kwargs):
        ...
        for at in Attribute.objects.all():
            if at.name in self.fields:
                self.fields[at.name].required = at.required
                if at.required and isinstance(self.fields[at.name], serializers.CharField):
                    self.fields[at.name].allow_blank = False
++              elif at.required and isinstance(self.fields[at.name], DateRestField):
++                  self.fields[at.name].allow_blank = False
```

Mais en fait on boucle ici sur les camps 'canoniques' : (à priori) ceux de RegistrationSerializer. Or comme cette classe ne contient n'a pas de champ date, cet ajout serait en fait du code mort.

Hors sujet, mais je ne comprend pas encore pourquoi ce patch n'a pas d'impact sur les tests CSV.
Par exemple, je m'attendais à ce qu'une date avec une chaîne blanche (ex: ' ') soit désormais acceptés, mais ce n'est pas le cas.

#6 - 25 septembre 2019 12:24 - Benjamin Dauvergne

L'import CSV n'utilise pas les champs DRF mais des champs des formulaires "classiques" et la classe de base pour les formulaire sur un utilisateur, dans l'idée qu'un CSV c'est un truc faisable à la main par un béotien plus proche d'un formulaire web que d'un document JSON (en gros c'est des chaînes comme dans un POST, c'est pas un truc qui peut être "null", ou "true" ou "false" ou un flottant ou un tableau ou un dico).

Ce qui est dommage c'est qu'on duplique beaucoup de choses aux passage (les sérialiseurs DRF sont ce que devraient être les Form django si on commençait from scratch aujourd'hui, c'est beaucoup plus puissant en terme de validation possible).

#7 - 03 octobre 2019 11:03 - Paul Marillonnet

Quelques modifs à mon avis nécessaires, notamment pour rendre le patch plus pythonique et plus compact (et un changement dans le message d'erreur de *may not*, qui est ambigu, pour *can't*, plus clair) :

```
diff --git a/src/authentic2/attribute_kinds.py b/src/authentic2/attribute_kinds.py
index 4d6dcdcf4..e16ce5c1d 100644
--- a/src/authentic2/attribute_kinds.py
+++ b/src/authentic2/attribute_kinds.py
@@ -58,12 +58,12 @@ class DateField(forms.DateField):

 class DateRestField(serializers.DateField):
     default_error_messages = {
-        'blank': _('This field may not be blank.'),
+        'blank': _('This field can\'t be blank.'),
     }

-    def __init__(self, **kwargs):
-        self.allow_blank = kwargs.pop('allow_blank', False)
-        self.trim_whitespace = kwargs.pop('trim_whitespace', True)
+    def __init__(self, allow_blank=False, trim_whitespace=True, **kwargs):
+        self.allow_blank = allow_blank
+        self.trim_whitespace = trim_whitespace
         super(DateRestField, self).__init__(**kwargs)

     def run_validation(self, data=empty):
diff --git a/src/authentic2/models.py b/src/authentic2/models.py
index 6f4b99911..11b1918f3 100644
--- a/src/authentic2/models.py
+++ b/src/authentic2/models.py
@@ -196,15 +196,10 @@ class Attribute(models.Model):
                 base_kwargs['allow_null'] = True
             # if not stated otherwise by the definition of the kind, string alike fields
             # accept blank values when not required
-            if (issubclass(field_class, serializers.CharField) and 'allow_blank' not in
-                    base_kwargs):
+            if (issubclass(field_class, (serializers.CharField, DateRestField))
+                    and 'allow_blank' not in base_kwargs):
                 base_kwargs['allow_blank'] = True
```

```
-            elif (issubclass(field_class, DateRestField) and 'allow_blank' not in
-                    base_kwargs):
-                base_kwargs['allow_blank'] = True
-        elif issubclass(field_class, serializers.CharField):
-            base_kwargs['allow_blank'] = False
-        elif issubclass(field_class, DateRestField):
+        elif issubclass(field_class, (serializers.CharField, DateRestField)):
             base_kwargs['allow_blank'] = False

         base_kwargs.update(kwargs)
```

#8 - 03 octobre 2019 11:11 - Paul Marillonnet

Ah oui, et quand même peut-être une ligne de commentaire au dessus de la définition de ta méthode DateRestField.run_validation pour indiquer qu'elle vient (i.e. est copiée à la virgule près) de rest_framework.fields.CharField.run_validation, par respect pour les auteurs du DRF :)

#9 - 03 octobre 2019 12:00 - Benjamin Dauvergne

Il ne faut pas suivre la proposition de Paul de changer le format de DateField.__init__ qui est :

```
    def __init__(self, format=empty, input_formats=None, *args, **kwargs):
```

pour respecter le principe de substitution de Liskov le *init* d'une sous-classe devrait avoir la forme suivante (s'il souhaite ajouter de nouveaux paramètres nommés) :

```
   def __init__(self, *args, **kwargs):
       self.my_new_kwarg = kwargs.pop('my_new_kwarg', None/False/True/etc..)
       super().__init__(*args, **kwargs)
```

#10 - 03 octobre 2019 12:05 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> pour respecter le principe de substitution de Liskov le *init* d'une sous-classe devrait avoir la forme suivante [...]

Et bien je ne connaissais pas. Je vais regarder ça, merci.

#11 - 03 octobre 2019 12:28 - Nicolas Roche

Paul, d'accord avec toi sur la forme, mais sur le fond :

- classe DateRestField est copiée/collée depuis rest_framework/fields.py::CharField
  Je pense que c'est mieux de garder le code le plus proche possible, notamment pour le message d'erreur qui sinon différera de celui des autres champs.
- classe Attribute est copiée/collée depuis le code de Benjamin.
  Idem je préfère garder le code en l'état pour ne pas altérer le 'git blame'

Qu'est-ce que tu en penses, est-ce que je corrige quand même ?

#12 - 03 octobre 2019 15:03 - Paul Marillonnet

- Statut changé de Solution proposée à Solution validée

Nicolas Roche a écrit :

> - classe DateRestField est copiée/collée depuis rest_framework/fields.py::CharField
>   Je pense que c'est mieux de garder le code le plus proche possible, notamment pour le message d'erreur qui sinon différera de celui des autres champs.

Ok, j'avais pas vu ça.

> - classe Attribute est copiée/collée depuis le code de Benjamin.
>   Idem je préfère garder le code en l'état pour ne pas altérer le 'git blame'

À mon avis c'est justifié de factoriser les appels à issubclass, mais c'est du détail, fais comme tu veux.

#13 - 03 octobre 2019 16:12 - Nicolas Roche

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

```
commit 05340b110b7d0cb9ba2a30a8b88fb6b22d7e56e6
Author: Nicolas ROCHE <nroche@entrouvert.com>
Date:   Tue Sep 24 19:17:20 2019 +0200

    api: extend DRF date field to accept empty string (#36365)
```

#14 - 04 octobre 2019 16:15 - Frédéric Péters

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

Fichiers

0001-api-extend-DRF-date-field-to-accept-empty-string-363.patch	8,96 ko 24 septembre 2019	Nicolas Roche
0001-api-extend-DRF-date-field-to-accept-empty-string-363.patch	8,84 ko 25 septembre 2019	Nicolas Roche