django-classifier Documentation

Release 0.2.1

Vadym Zakovinko

Nov 15, 2016

Contents

1 Content **3**
- 1.1 Installation . . . 3
- 1.2 Tutorial . . . 4
- 1.3 `classifier` package . . . 6

2 Indices and tables **9**

Python Module Index **11**

Flexible constructor to create dynamic list of heterogeneous properties for some kind of entity. This set of helpers useful to create properties like contacts or attributes for describe car/computer/etc.

You must create your own models for each kind of entity but it gives more flexibility, more simpler and allow to easily change default behavior.

read *Tutorial* to see all picture of idea.

CHAPTER 1

Content

1.1 Installation

1.1.1 Requirements

- Python 2.7, 3.4, 3.5
- Django 1.8, 1.9, 1.10

1.1.2 Official releases

Official releases are available from PyPI.

Download the .zip distribution file and unpack it. Inside is a script named `setup.py`. Enter this command:

```
python setup.py install
```

...and the package will install automatically.

or just install with help of pip:

```
pip install django-classifier
```

1.1.3 Development version

Alternatively, you can get the latest source from our git repository:

```
git clone http://github.com/django-stars/django-classifier.git django-classifier
```

Add `django-classifier/classifier` folder to PYTHONPATH.

You can verify that the application is available on your PYTHONPATH by opening a Python interpreter and entering the following commands:

```
>> import classifier
>> classifier.VERSION
(0, 2)
```

Caution: The development version may contain bugs which are not present in the release version and introduce backwards-incompatible changes.

1.2 Tutorial

1.2.1 Installation

Simplest way:

```
pip install django-classifier
```

see more at *Installation*.

1.2.2 Configuration

Add *classifier* to INSTALLED_APPS in your setting.py file:

```
INSTALLED_APPS = (
    'djangp.contrib.auth',
    # ...
    'classifier',
)
```

1.2.3 Set up your models

You need to create two models for classifier structure and one for data.

First it's basic model without extra data inherited from *ClassifierAbstract*:

```
from classifier.models import ClassifierAbstract

class ContactClassifier(ClassifierAbstract):
    pass
```

Second model must contain ForeignKey to model that we created before and inherited from *ClassifierLabelAbstract*:

```
from classifier.models import ClassifierLabelAbstract

class ContactClassifierLabel(ClassifierLabelAbstract):
    classifier = models.ForeignKey(ContactClassifier, related_name='labels')
```

Model for data pretty simple in general: ForeignKey to label model (ContactClassifierLabel in tutorial) and CharField for value:

```
from django.conf import settings
from django.db import models

class Contact(models.Model):
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
```

```
        related_name='contacts'
    )
    kind = models.ForeignKey(ContactClassifier)
    value = models.CharField(max_length=500)
```

Then make and run migrations and you are ready to use `django-classifier`:

```
python manage.py makemigrations
python manage.py migrate
```

1.2.4 Add some data to classifier models

Lets add some data to `ContactClassifier` and `ContactClassifierLabel`:

```
contact_classifier1 = ContactClassifier.objects.create(
    kind='phone',
    value_type=ContactClassifier.TYPES.STRING,
    value_validator=r'^\+\d{12}$',
    only_one_required=False,
)
contact_classifier1.labels.create(label='Mobile')
contact_classifier1.labels.create(label='Work')
contact_classifier1.labels.create(label='Home')

contact_classifier2 = ContactClassifier.objects.create(
    kind='im',
    value_type=ContactClassifier.TYPES.STRING,
    value_validator=None,
    only_one_required=False,
)
contact_classifier2.labels.create(label='Skype')
contact_classifier2.labels.create(label='Jabber')
```

1.2.5 Create form for validation

To validate values based on classifier record, for this `django-classifier` has *`ClassifierFormMixin`*:

```
from django import forms
from classifier.forms import ClassifierFormMixin
from .models import Contact

class ContactForm(ClassifierFormMixin, forms.ModelForm):
    CLASSIFIER_VALUE_FIELD = 'value'

    class Meta:
        model = Contact
        fields = ('user', 'kind', 'value', )
```

You have to specify *`CLASSIFIER_VALUE_FIELD`* attribute to identify name of value field to attach validation based on classifier record.

1.2.6 Playing with formset

classifier provide own `FormSet` class *ClassifierFormSet* that will add extra forms for mandatory records and validate if all of them are filled in:

```
from django.contrib.auth import get_user_model
from django.forms import modelformset_factory
from classifier.formsets import ClassifierFormSet
from .forms import UserForm, ContactForm
from .models import Contact

ContactFormSet = modelformset_factory(
    Contact,
    formset=ClassifierFormSet,
    form=ContactForm
)

user = get_user_model().objects.create('login', 'password')
contact_formset = ContactFormSet(queryset=user.contacts.all())

print(len(contact_formset.forms))
print(contact_formset.forms[0].initial)
```

For now any of classifier records are not marked as required and formset has only one blank forms as default in Django.

But if you will mark both classifiers as *only_one_required* you will have two forms with prepopulated labels (*first available label*):

```
ContactClassifier.objects.all().update(only_one_required=True)

contact_formset = ContactFormSet(queryset=user.contacts.all())
print(len(contact_formset.forms))
print(contact_formset.forms[0].initial)
print(contact_formset.forms[1].initial)
```

If you will mark all labels as *required* then you will have 5 forms by default and all of them will be required:

```
ContactClassifier.objects.all().update(only_one_required=False)
ContactClassifierLabel.objects.all().update(required=True)

contact_formset = ContactFormSet(queryset=user.contacts.all())
print(len(contact_formset.forms))
```

1.3 `classifier` package

1.3.1 `classifier.models`

`ClassifierAbstract`

class `classifier.models.`**`ClassifierAbstract`**(**args*, ***kwargs*)

Base class to create classifier models. Will provide base data and functions like validation and `to_python` (convert from string to real type).

In simplest case should be created model just inherited from this abstract model without extra code.

`kind` - custom identificator for classifier type (like: phone) `value_type` - expected type of value (like: string) `value_validator` - regex to validate extered value (like: +d{12}) `only_one_required` - checkmark to make one on available lables required

Supported types: `int`, `float`, `string`, `boolean`, `date`, `datatime`.

Labels with kind give posibility to create one type of record with different names, like kind is *phone* and available lables are "Mobile", "Home", "Work" etc.

kind
custom identificator for classifier type (like: phone)

value_type
expected type of value (like: string)

value_validator
regex to validate extered value (like: +d{12})

only_one_required
checkmark to make one on available lables required

to_python(*value*)
run convertor from string to type in `value_type` field

ClassifierLabelAbstract

class classifier.models.**ClassifierLabelAbstract**(**args*, ***kwargs*)
Base model class to define several human readable names for each classifier kind.

label
human redable label for data

required
checkmark to make label required

get_classifier_instance()

Returns instance of related classifier

classmethod get_classifier_related_field()

Returns field related to model inherited from ClassifierAbstract.

Raises **ClassifierModelNotFound** – if related field wasn't found

> **Caution:** not field name

classmethod get_classifier_model()

Returns related model inherited from ClassifierAbstract

1.3.2 classifier.formsets

ClassifierFormSet

class classifier.formsets.**ClassifierFormSet**(**args*, ***kwargs*)

add_required_to_extra()
Method create extra forms for required records from classifier with selected needed kinds by default.

classifier_label_model
Property return model inherited from *ClassifierLabelAbstract* and used for one of field in model for this formset.

Returns model inherited from *ClassifierLabelAbstract*

Raises **ClassifierLabelModelNotFound** – field can not be found

classifier_label_related_fieldname
Return name of field related to model inherited from *ClassifierLabelAbstract*.

validate_required()
Validate if all required records are filled in

Raises **django.core.exceptions.ValidationError** – if one or mode records are absent

1.3.3 classifier.forms

ClassifierFormMixin

class classifier.forms.**ClassifierFormMixin**(**args*, ***kwargs*)
Formset form mixin to enable validation for value connected to classifier.

CLASSIFIER_VALUE_FIELD = None
Name of field for value used in relation with classifier

classifier_label_fieldname

Returns Return name of field that is relation to end model inherited from *ClassifierLabelAbstract*

Raises **ClassifierLabelModelNotFound** – if field can not be found

classifier_label_model

Returns end model inherited from *ClassifierLabelAbstract* used in current form

Raises **ClassifierLabelModelNotFound** – if related field wasn't found

setup_value_validators()
Attach validator for value field specified in *CLASSIFIER_VALUE_FIELD*

Raises **NoValueFieldNameSpecified** – if *CLASSIFIER_VALUE_FIELD* is blank

validate_value_field()
Validate value based on classifier record.

Will be attathed to right field by call *setup_value_validators()* in __init__()

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

C

classifier, 6
classifier.forms, 8
classifier.formsets, 7
classifier.models, 6

A

add_required_to_extra() (classifier.formsets.ClassifierFormSet method), 7

C

classifier (module), 6
classifier.forms (module), 8
classifier.formsets (module), 7
classifier.models (module), 6
classifier_label_fieldname (classifier.forms.ClassifierFormMixin attribute), 8
classifier_label_model (classifier.forms.ClassifierFormMixin attribute), 8
classifier_label_model (classifier.formsets.ClassifierFormSet attribute), 8
classifier_label_related_fieldname (classifier.formsets.ClassifierFormSet attribute), 8
CLASSIFIER_VALUE_FIELD (classifier.forms.ClassifierFormMixin attribute), 8
ClassifierAbstract (class in classifier.models), 6
ClassifierFormMixin (class in classifier.forms), 8
ClassifierFormSet (class in classifier.formsets), 7
ClassifierLabelAbstract (class in classifier.models), 7

G

get_classifier_instance() (classifier.models.ClassifierLabelAbstract method), 7
get_classifier_model() (classifier.models.ClassifierLabelAbstract class method), 7
get_classifier_related_field() (classifier.models.ClassifierLabelAbstract class method), 7

K

kind (classifier.models.ClassifierAbstract attribute), 7

L

label (classifier.models.ClassifierLabelAbstract attribute), 7

O

only_one_required (classifier.models.ClassifierAbstract attribute), 7

R

required (classifier.models.ClassifierLabelAbstract attribute), 7

S

setup_value_validators() (classifier.forms.ClassifierFormMixin method), 8

T

to_python() (classifier.models.ClassifierAbstract method), 7

V

validate_required() (classifier.formsets.ClassifierFormSet method), 8
validate_value_field() (classifier.forms.ClassifierFormMixin method), 8
value_type (classifier.models.ClassifierAbstract attribute), 7
value_validator (classifier.models.ClassifierAbstract attribute), 7