django-formaldehyde Documentation

Release 0.4

Lucio Asnaghi

June 15, 2015

Contents

1 Getting started **3**

2 Configuration **5**

3 Classes and Mixins **7**
- 3.1 FieldsetFormMixin . . . 7
- 3.2 ReadonlyFormMixin . . . 8
- 3.3 StripWhitespaceFormMixin . . . 8

4 Support **9**

5 Changelog **11**
- 5.1 v0.4 (unreleased) . . . 11
- 5.2 v0.3 (2015-01-30) . . . 11
- 5.3 v0.2 (2015-01-30) . . . 11
- 5.4 v0.1 (2015-01-21) . . . 11

Django Formaldehyde is a small reusable unobtrusive Django app that leverage forms.

- Home page: https://github.com/kunitoki/django-formaldehyde
- Documentation: http://django-formaldehyde.readthedocs.org
- Pypi: https://pypi.python.org/pypi/django-formaldehyde
- Example app on Github: https://github.com/kunitoki/django-formaldehyde/example
- Changelog: Changelog.rst
- License: The MIT License (MIT)
- Supports: Django 1.6, 1.7, 1.8 - Python 2.7, 3.2, 3.3, 3.4

CHAPTER 1

Getting started

1. You can get Django Formaldehyde by using pip:

```
pip install django-formaldehyde
```

2. You will need to add the `'formaldehyde'` application to the `INSTALLED_APPS` setting of your Django project `settings.py` file.:

```
INSTALLED_APPS = (
    ...

    'formaldehyde',
)
```

CHAPTER 2

Configuration

It is possible to configure by using in `settings.py`.:

```
FORMALDEHYDE_SETTINGS = {
    'GRID_COLUMN_NUMBER': 12,
    'DEFAULT_FIELD_COLUMN_SIZE': 1,
    'DEFAULT_LABEL_COLUMN_SIZE': 1,
    'FIRST_LABEL_COLUMN_SIZE': 2
}
```

CHAPTER 3

Classes and Mixins

django-formaldehyde provides a series of django.forms.Form mixins to help the handling of forms in applications.

3.1 FieldsetFormMixin

The mixin add fieldsets with fieldlines to forms. It also add the concept of layout of the fields, so it is possible to place fields and labels in the grid space. Everything is used inside an inner class called MetaForm:

```
from django import forms
from formaldehyde.fieldsets import FieldsetFormMixin

class ExampleFieldsetForm(FieldsetFormMixin, forms.Form):
    first_name = forms.CharField(label='First name', max_length=100)
    last_name = forms.CharField(label='Last name', max_length=100)
    nick_name = forms.CharField(label='Nick name', max_length=100)
    avatar = forms.IntegerField()
    photo = forms.ImageField()

    class Meta:
        fields = '__all__'

    class MetaForm:
        fieldsets = (
            (None, {
                'fields': (
                    ('first_name', 'last_name'),
                    'nick_name',
                ),
                'layout': (
                    (4, 6),
                    2
                ),
                'labels': (
                    (2, 1),
                    1
                ),
                'classes': 'form-control'
            }),
            ('Social', {
                'fields': (
                    'avatar',
                    'photo'
```

```
                )
            ),
        )
```

An example template (for bootstrap grid system) for rendering the form looks like:

```
{% if form.fieldsets %}
    {% for fieldset in form.fieldsets %}
        <fieldset class="{{ fieldset.classes }}">
            {% if fieldset.legend %}
                <legend>{{ fieldset.legend }}</legend>
            {% endif %}
            {% if fieldset.description %}
                <p class="description">{{ fieldset.description }}</p>
            {% endif %}
            {% for fieldline in fieldset %}
                <div class="form-group">
                {% for field, layout in fieldline %}
                    {% if field.is_hidden %}
                        {{ field }}
                    {% else %}
                        <label class="col-sm-{% if forloop.first %}2{% else %}1{% endif %} control-la
                            {{ field.label }}
                        </label>
                        <div class="col-sm-{{ layout|default:10 }}">
                            {{ field }}
                        </div>
                    {% endif %}
                {% endfor %}
                </div>
            {% endfor %}
        </fieldset>
    {% endfor %}
{% else %}
    {% for hidden in form.hidden_fields %}
        {{ hidden }}
    {% endfor %}
    {% for field in form.visible_fields %}
        <div class="form-group{% if field.errors %} has-error{% endif %}">
            <label class="col-sm-{% if forloop.first %}2{% else %}1{% endif %} control-label">
                {{ field.label }}
            </label>
            <div class="col-sm-{{ field_size|default:10 }}">
                {{ field }}
            </div>
        </div>
    {% endfor %}
{% endif %}
```

3.2 ReadonlyFormMixin

3.3 StripWhitespaceFormMixin

CHAPTER 4

Support

- Github: Use django-formaldehyde github issues, if you have any problems using Django Formaldehyde.

CHAPTER 5

Changelog

Only important changes are mentioned below.

5.1 v0.4 (unreleased)

- refactored field line iteration (backward incompatible)
- fieldsets in MetaForm take both 'labels' and 'layout' to specify labels size in grid
- fixed a problem with immutable MergeDict in StripWhitespaceFormMixin
- updated docs

5.2 v0.3 (2015-01-30)

- fieldsets in MetaForm can be avoided if preferred

5.3 v0.2 (2015-01-30)

- Fixed problem with assertion on django.forms.ModelForm

5.4 v0.1 (2015-01-21)

- First version released