Django Fancy Cronfield Documentation

Release 0.2.0

Saeed Salehian

December 16, 2015

Contents

CHAPTER 1

Overview

Django fancy cronfield is a nice and customizable cron field.

Developers who want to learn more about Django Fancy Cronfield, as well as how to install, use and customise it for their own projects will can refer to Tutorials and Reference sections.

CHAPTER 2

Join us online

You can find us on our mailing list.

CHAPTER 3

Features

- Cron widget providing nice gentle select UI
- Cron format validation
- Custom django field
- Ability to specify a daily run limit

CHAPTER 4

Release Notes

This document refers to version 0.2.0

CHAPTER 5

Table of contents

5.1 Tutorials

The pages in this section of the documentation are aimed at the newcomer. They're designed to help you get started quickly, and show how easy it is to work with it as a developer who wants to customise it and get it working according to their own requirements.

5.1.1 Installation

```
pip install django-fancy-cronfield
```

5.1.2 Basic usage

Use it like any regular model field:

```
from django.db import models

from fancy_cronfield.fields import CronField


class MyModel(models.Model):
    timing = CronField()
```

5.1.3 Field Options

Django fancy cronfield extend Field, so all options available at Field can be used with CronField as well. Here is a list of kwargs which CronField has added or altered:

- *max_length*: is set to 120 by default, however you can override it at will.
- *daily_limit*: Specifies maximum value which cron frequency per day can be. *None* means no limit.

Please see Fields for an inside look into CronField.

5.1.4 Widget Options

Django fancy cronfield comes with a useful custom widget which provides a gentle select UI for specifying cron strings at administration panel. However you can customize the widget UI behaviour by passing *options* dictionary to `CronWidget` constructor.

For example, if you want to customize the `CronWidget` for the timing `CronField` of `Schedule` to use <select> instead of the default gentle select, you can override the field's widget and pass your desired options:

```
from django import forms
from fancy_cronfield.widgets import CronWidget

from myapp.models import Schedule


class ScheduleForm(forms.ModelForm):
    class Meta:
        model = Schedule
        fields = ('name', 'timing')
        widgets = {
            'timing': CronWidget(
                attrs={'class': 'special'},
                options={'use_gentle_select': True}
            ),
        }
```

Note: Please note that *options* parameter differs from *attrs* which is used to specify html attributes. *options* are limited to a list of predefined items which are described below.

Here are the list of options that you can use to customize UI behaviour:

use_gentle_select

Default `True` means using gentle select UI by default

Boolean that determines if the widget should use gentle select UI or simple select input.

allow_multiple_all

Default `False`

Boolean that decides if the widget should allow multiple selection on all cron parts. When this is *True*, the user can select multiple values for each cron part. *False* meant that cron parts are not forced to allow multiple selection, and each cron part's individual option will decide about the behaviour.

allow_multiple_dom

Default `True`

Boolean that decides if the widget should allow multiple selection on *day of month* cron part.

allow_multiple_month

Default `True`

Boolean that decides if the widget should allow multiple selection on *month* cron part.

allow_multiple_dow

Default `True`

Boolean that decides if the widget should allow multiple selection on *day of week* cron part.

allow_multiple_hour

Default `True`

Boolean that decides if the widget should allow multiple selection on *hour* cron part.

allow_multiple_minute

Default `True`

Boolean that decides if the widget should allow multiple selection on *minute* cron part.

Below you can find an example options dictionary, these options indicates that the widget should use gentle select to render itself, allow multiple selection on day of month, month and day of week. However multiple selection is not allowed for hour and minute part.

Example:

```
options = {
    'use_gentle_select': True,
    'allow_multiple_all': False,
    'allow_multiple_dom': True,
    'allow_multiple_month': True,
    'allow_multiple_dow': True,
    'allow_multiple_hour': False,
    'allow_multiple_minute': False
}
```

Note: There might be a case where you need to use the default `TextInput` widget instead of `CronWidget`. It could easily be done by overriding the field's widget in ModelForm.

Please see Widgets for an inside look into CronWidget.

5.2 Reference

Technical reference material.

5.2.1 Fields

fancy_cronfield.fields

class `fancy_cronfield.fields.`**`CronField`**(**args*, ***kwargs*)

Custom cron field which extends CharField and does extra initializations:

- Enforcing maximum length of 120 by default, using *max_length*
- Checks if daily limit is provided via *daily_limit* option
- Appends *CronValidator* to field validators

`formfield`(***kwargs*)

Returns a django.forms.Field instance for this database field which uses CronWidget for supporting gentle select UI.

Passing max_length to widgets.CronWidget means that the value's length will be validated twice. This is considered acceptable since we want the value in the form field (to pass into widget for example).

Parameters **`kwargs`** – dict, form field key word arguments

Returns django.forms.Field instance

get_internal_type()
It is most similar to Django CharField class

Returns string *CharField*

get_prep_value(*value*)

to_python(*value*)

5.2.2 Widgets

fancy_cronfield.widgets

class fancy_cronfield.widgets.**CronWidget**(*attrs=None*, *options=None*)
CronWidget class providing gentle select UI for cron fields

class Media

css = {'all': ('fancy_cronfield/css/cronfield.min.css', 'fancy_cronfield/css/jquery-cron.min.css', 'fancy_cronfield/css/j

js = ('fancy_cronfield/js/jquery-1.4.1.min.js', 'fancy_cronfield/js/jquery-cron.min.js', 'fancy_cronfield/js/jquery-gen

CronWidget.**input_type = 'hidden'**

CronWidget.**media**

5.2.3 Validators

fancy_cronfield.validators

class fancy_cronfield.validators.**CronValidator**(*limit_value*, *message=None*)
Cron format validator which does the following actions:

•Ensures that the cron string is a valid cron format

•Ensures that the cron frequency per day is less than *limit_value*

clean(*value*)
Ensures that the given value is a valid cron format and strips it.

Parameters **value** – cron string

Returns Stripped value string

code = 'cron'

error_messages = {'invalid_cron': u'Ensure that your selected timing is valid.'}

message = u'Ensure the timing runs at most %(limit_value)s times per day. It runs %(show_value)s times per day.'

5.3 Release notes & upgrade information

Some versions of django fancy cronfield present more complex upgrade processes than others, and some **require** you to take action. It is strongly recommended to read the release notes carefully when upgrading.

It goes without saying that you should **backup your database** before embarking on any process that makes changes to your database.

5.3.1 0.1 release notes

What's new in 0.1

Cronfield

This version introduces a new custom django field for use of cron strings in django models.

Cronfield introduces an new parameter named *daily_limit* which can be used to limit daily frequency of the given cron string.

Python Support

The Django Fancy Cronfield supports python 2.6, 2.7, 3.3 and 3.4.

Django Support

The Django Fancy Cronfield supports django 1.5, 1.6, 1.7 and 1.8.

5.4 Indices and tables

- genindex
- modindex
- search

f

A

C

E

F

G

I

J

M

T

U