django-siteflags Documentation

Release 0.5.0

Igor 'idle sign' Starikov

Sep 17, 2018

Contents

https://github.com/idlesign/django-siteflags

CHAPTER 1

Description

Reusable application for Django allowing users to flag/bookmark site objects

So you want a user to be able to put some flags on certain site entities.

Let's say you need a kind of bookmark powered service, or a site where content is flagged and moderated, or a simplified rating system or something similar.

CHAPTER 2

Requirements

1. Python 2.7+, 3.4+
2. Django 1.7+
3. Django Auth contrib enabled
4. Django Admin contrib enabled (optional)

CHAPTER 3

Table of Contents

3.1 Quickstart

Note: Add the **siteflags* application to INSTALLED_APPS in your settings file (usually 'settings.py').

Let's suppose we want our users to report fake articles.

Inherit you model from **siteflags.models.ModelWithFlag** and you're almost done.

myapp/models.py:

```
from siteflags.models import ModelWithFlag

class Article(ModelWithFlag):

    FLAG_FAKE = 10

    ... # Some model fields here.
```

myapp/views.py:

```
from django.shortcuts import get_object_or_404
from .models import Article


def article_details(request, id):

    article = get_object_or_404(Article, pk=id)

    ...

    # Now a user reports this article as a fake.
    article.set_flag(request.user, note='Hi, I found this article a fake!',␣
↪status=Article.FLAG_FAKE)
```

(continues on next page)

(continued from previous page)

```
    ...
```

And that's how it's done.

3.2 ModelWithFlag Model

siteflags.models.ModelWithFlag is practically all that's needed for flagging.

3.2.1 Methods

get_flags_for_types(mdl_classes, [user=None[, status=None[, allow_empty=False]]]):
Returns a dictionary with flag objects associated with the given model classes (types). The dictionary is indexed by model classes. Each dict entry contains a list of associated flag objects.

Parameters

- **mdl_classes** (*list*) – Classes objects (types) list to get flags for.
- **user** (*User*) – Optional user filter
- **status** (*int*) – Optional status filter
- **allow_empty** (*bool*) – Include results for all given types, even those without associated flags.

get_flags_for_objects(objects_list, [user=None[, status=None]]):
Returns a dictionary with flag objects associated with the given objects. The dictionary is indexed by objects IDs. Each dict entry contains a list of associated flag objects.

Parameters

- **QuerySet objects_list** (*list,*) – Homogeneous objects list to get flags for.
- **user** (*User*) – Optional user filter
- **status** (*int*) – Optional status filter

get_flags([user=None[, status=None]]):
Returns flags for the object optionally filtered by user and/or status.

Parameters

- **user** (*User*) – Optional user filter
- **status** (*int*) – Optional status filter

set_flag(user[, note=None[, status=None]]):
Flags the object.

Parameters

- **user** (*User*) –
- **note** (*str*) – User-defined note for this flag.
- **status** (*int*) – Optional status integer (the meaning is defined by a developer).

remove_flag([user=None[, status=None]]):
Removes flag(s) from the object.

Parameters

- **user** (*User*) – Optional user filter
- **status** (*int*) – Optional status filter

is_flagged([user=None[, status=None]]):
Returns boolean whether the objects is flagged by a user.

Parameters

- **user** (*User*) –
- **status** (*int*) – Optional status filter

3.2.2 Customization

SiteFlags allows you to customize Flags model.

1. Define your own *flag* model inherited from *FlagBase*.

2. Now when *models.py* in your application has the definition of a custom flags model, you need to instruct Django to use it for your project instead of a built-in one:

```
# Somewhere in your settings.py do the following.
# Here `myapp` is the name of your application, `MyFlag` is the names of your␣
↪customized model.

SITEFLAGS_FLAG_MODEL = 'myapp.MyFlag'
```

3. Run *manage.py syncdb* to install your customized models into DB.

CHAPTER 4

Get involved into django-siteflags

Submit issues. If you spotted something weird in application behavior or want to propose a feature you can do that at https://github.com/idlesign/django-siteflags/issues

Write code. If you are eager to participate in application development, fork it at https://github.com/idlesign/django-siteflags, write your code, whether it should be a bugfix or a feature implementation, and make a pull request right from the forked project page.

Translate. If want to translate the application into your native language use Transifex: https://www.transifex.com/projects/p/django-siteflags/.

Spread the word. If you have some tips and tricks or any other words in mind that you think might be of interest for the others — publish them.

CHAPTER 5

Also

If the application is not what you want for content flagging/bookmarking, you might be interested in considering other choices — https://www.djangopackages.com/grids/g/bookmarking/