django-siteflags Documentation

Release 1.3.0

Igor 'idle sign' Starikov

Feb 04, 2022

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 3.6+
2. Django 2.0+
3. Django Auth contrib enabled
4. Django Admin contrib enabled (optional)

CHAPTER 3

Table of Contents

3.1 Quickstart

Note: Do not forget to add the `siteflags` application to `INSTALLED_APPS` in your settings file (usually `settings.py`) and apply migrations.

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

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

3.1.1 myapp/models.py

```
from siteflags.models import ModelWithFlag

class Article(ModelWithFlag):

    FLAG_FAKE = 10
    """Let's suppose we have several flag types.
    And this is a flag status for "fake" flag type.

    """

    FLAG_BOOKMARK = 20
    """And this is a flag status for "bookmark" flag type."""

    ...  # Some model fields here.

    # Now we may want define fake-related helper methods.

    def fake_mark_add(self, user, note):
        return self.set_flag(user, note=note, status=self.FLAG_FAKE)
```

(continues on next page)

(continued from previous page)

```
    def fake_mark_remove(self, user):
        return self.remove_flag(user, status=self.FLAG_FAKE)

    def fake_mark_check(self, user):
        return self.is_flagged(user, status=self.FLAG_FAKE)

    ...  # Maybe also some helper methods for FLAG_BOOKMARK.
```

3.1.2 myapp/views.py

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


def article_details(request, article_id):

    article = get_object_or_404(Article, pk=article_id)

    user = request.user
    # Let's suppose we have here only logged in users.

    post = request.POST

    if post.get('fake_set'):
        # Now a user reports this article as a fake.
        article.fake_mark_add(user, note=post.get('fake_message'))

    elif post.get('fake_remove'):
        # Or he removes a fake flag.
        article.fake_mark_remove(user)

    is_fake = article.fake_mark_check(user)
    # This you may want to pass into a template to show flag state.

    ...  # Maybe also some handling for FLAG_BOOKMARK.

    # That's how we get all article flags (any type/status)
    # for the current user.
    all_flags = article.get_flags(user)

    ...  # Maybe render a template here.
```

There are even more generic API methods:

```
from siteflags.models import ModelWithFlag

# We can find flags of any type for various objects.
# Let's pretend we also 'article', 'video' and 'image' objects
# available in the current scope.
flags = ModelWithFlag.get_flags_for_objects([article, video, image])

# We can also find flags of any type by type.
# Let's also prefetch Article objects (with_objects=True).
flags = Article.get_flags_for_type(with_objects=True)
```

(continues on next page)

(continued from previous page)

```
# And that's practically would be the same as in 'all_flags'
# of the above mentioned view.

for flag in flags:
    # Since we've prefetched the linked objects with our flags
    # we can access article properties without additional DB hits.
    print(f'article: {flag.linked_object.id}')
```

Note: You can also customize `Flag` model by inheriting from `siteflags.models.FlagBase` and setting `SITEFLAGS_FLAG_MODEL` in your `settings.py`, for example:

```
SITEFLAGS_FLAG_MODEL = 'myapp.MyFlag'
```

And that's how it's done.

Warning: If you use a custom model and override `Meta`, be sure to inherit it from `FlagBase.Meta`. Otherwise you may miss `unique_together` constraints from the base class.

3.2 ModelWithFlag Model

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

3.2.1 Methods

get_flags_for_type([mdl_classes=None, [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 makemigrations` and `manage.py migrate` 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/