django-siteprefs Documentation

Release 1.2.3

Igor 'idle sign' Starikov

Dec 18, 2021

Contents

1 Requirements **3**

2 Table of Contents **5**

2.1 Getting started 5

2.2 Preferences registration 6

2.3 Settings 7

3 Get involved into django-siteprefs **9**

4 Also **11**

http://github.com/idlesign/django-siteprefs

Reusable app for Django introducing site preferences system

django-siteprefs allows Django applications settings to come alive.

CHAPTER 1

Requirements

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

CHAPTER 2

Table of Contents

2.1 Getting started

- Add the **siteprefs** application to INSTALLED_APPS in your settings file (usually 'settings.py').
- Use > `python manage.py migrate` command to install apps tables int DB.

2.1.1 Quick example

Let's suppose we created `MYAPP` application and now create `settings.py` file for it:

```
from django.conf import settings

ENABLE_GRAVATARS = getattr(settings, 'MYAPP_ENABLE_GRAVATARS', True)
ENABLE_MAIL_RECOVERY = getattr(settings, 'MYAPP_ENABLE_MAIL_RECOVERY', True)
ENABLE_MAIL_BOMBS = getattr(settings, 'MYAPP_ENABLE_MAIL_BOMBS', False)
SLOGAN = "I'm short and I'm tall // I'm black and I'm white"
PRIVATE_SETTING = 'Hidden'


if 'siteprefs' in settings.INSTALLED_APPS:  # Respect those users who doesn't have␣
↪siteprefs installed.

    from siteprefs.toolbox import preferences

    with preferences() as prefs:

        prefs(  # Now we register our settings to make them available as siteprefs.
            # First we define a group of related settings, and mark them non-static␣
↪(editable).
            prefs.group('Mail settings', (ENABLE_MAIL_RECOVERY, ENABLE_MAIL_BOMBS),␣
↪static=False),
            SLOGAN,  # This setting stays static non-editable.
```

(continues on next page)

(continued from previous page)

```
            # And finally we register a non-static setting with extended meta for␣
↪Admin.
            prefs.one(
                ENABLE_GRAVATARS,
                verbose_name='Enable Gravatar service support', static=False,
                help_text='This enables Gravatar support.'),
        )
```

From now on you can view (and edit) your preferences with Django Admin interface.

Access your settings as usual, all changes made to preferences with Admin interface will be respected:

```
from .settings import ENABLE_MAIL_BOMBS

def bombing():
    if ENABLE_MAIL_BOMBS:
        print('booooom')
```

And mind that we've barely made a scratch of **siteprefs**.

2.2 Preferences registration

siteprefs has several helpers to ease application settings registration.

All of them reside in *siteprefs.toolbox* module. Let's go one by one:

- *register_prefs(*args, **kwargs)*

  The main way to register your settings. Expects preferences as **args** and their options as **kwargs**:

  ```
  register_prefs(
      MY_OPT_1,
      MY_OPT_2,
      MY_OPT_3,
      category='All the settings'  # This will group all the settings into one␣
  ↪category in Admin.
  )
  ```

- *pref_group(group_title, prefs, **kwargs)*

  This allows preferences grouping. Expects a group title, a list of preferences and their options as **kwargs**:

  ```
  register_prefs(
      MY_OPT_1, MY_OPT_2,
      pref_group('My options group 1', (MY_OPT_3, MY_OPT_4), static=False),
      pref_group('My options group 2', (MY_OPT_5, pref(MY_OPT_6, verbose_name='My 6
  ↪')')),
  )
  ```

- *pref(preference, **kwargs)*

  Used to mark a preference. Expects a preference and its options as **kwargs**:

  ```
  register_prefs(
      MY_OPT_1, MY_OPT_2,
      pref(MY_OPT_3, verbose_name='My third option', static=False),
  ```

(continues on next page)

(continued from previous page)

```
    pref(MY_OPT_4, verbose_name='Fourth', help_text='My fourth option.'),
)
```

The functions mentioned above are available through a shortcut `preferences()` context manager as mentioned in the Quickstart.

2.2.1 Options accepted by prefs

These are the options accepted as **kwargs** by **siteprefs** helpers:

- `static`

  Flag to mark a preference editable from Admin - static are not editable. True by default.

- `readonly`

  Flag to mark an [editable] preference read only for Admin. False by default.

- `field`

  Field instance (from django.db.models, e.g. `BooleanField()`) to represent a sitepref in Admin.

  None by default. If None, **siteprefs** will try to determine an appropriate field type for a given preference value type.

- `category`

  Category name to group a sitepref under. None by default.

- `verbose_name`

  Preference name to render in Admin.

  None by default. If None, a name will be deduces from preference variable name.

- `help_text`

  Hint text to render for a preference in Admin. Empty by default.

2.3 Settings

Some aspects of **siteprefs** could be tuned in *settings.py* of your project.

2.3.1 SITEPREFS_EXPOSE_MODEL_TO_ADMIN

Toggles internal preferences model showing up in the Admin. Default: true.

2.3.2 SITEPREFS_DISABLE_AUTODISCOVER

Disables preferences autodiscovery on Django apps registry ready. Default: false.

2.3.3 SITEPREFS_MODULE_NAME

Module name used by siteprefs.toolbox.autodiscover_siteprefs() to find preferences in application packages.

CHAPTER 3

Get involved into django-siteprefs

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-siteprefs/issues

Write code. If you are eager to participate in application development, fork it at https://github.com/idlesign/django-siteprefs, 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-siteprefs/.

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 4

Also

Consider more applications on the subject — https://www.djangopackages.com/grids/g/live-setting/