django-siteprefs Documentation

Release 0.6.3

Igor 'idle sign' Starikov

Mar 30, 2017

Contents

1 Requirements **3**

2 Table of Contents **5**

2.1 Getting started 5

2.2 Preferences registration 6

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 2.7+ or 3.3+
2. Django 1.4+
3. Django Auth contrib enabled
4. Django Admin contrib enabled (optional)
5. South 1.0+ (for automatic DB migrations; not required for Django 1.7+)

CHAPTER 2

Table of Contents

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 (`> python manage.py syncdb` for Django <1.7)
- For Django <1.7 add preferences `autodiscover` function call into urls.py file:

```
from siteprefs.toolbox import autodiscover_siteprefs

autodiscover_siteprefs()
```

Warning: Those, who are using South <1.0 for migrations, add this into settings file:

```
SOUTH_MIGRATION_MODULES = {
    'siteprefs': 'siteprefs.south_migrations',
}
```

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 patch_locals, register_prefs, pref, pref_group

    patch_locals()  # That's bootstrap.

    register_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).
        pref_group('Mail settings', (ENABLE_MAIL_RECOVERY, ENABLE_MAIL_BOMBS),␣
↪static=False),
        SLOGAN,  # This setting stays static non-editable.
        # And finally we register a non-static setting with extended meta for Admin.
        pref(ENABLE_GRAVATAR_SUPPORT, 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**.

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

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.

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/