Wagtail Settings Documentation

Release 0.3.3

Tim Heap

April 07, 2015

Contents

Contents:

CHAPTER 1

Setup

`wagtailsettings` is compatible with Wagtail 1.0 and higher, Django 1.7 and higher, and runs on Python 2.7 or Python 3.4.

1. Install `wagtailsettings` using pip:

```
pip install wagtailsettings
```

2. Add it to your `INSTALLED_APPS`:

```
INSTALLED_APPS += [
    'wagtailsettings',
]
```

CHAPTER 2

Defining settings

Create a model that inherits from `BaseSetting`, and register it using the `register_setting` decorator:

```
from wagtailsettings import BaseSetting, register_setting

@register_setting
class SocialMediaSettings(BaseSetting):
    facebook = models.URLField(
        help_text='Your Facebook page URL')
    instagram = models.CharField(
        max_length=255, help_text='Your Instagram username, without the @')
    trip_advisor = models.URLField(
        help_text='Your Trip Advisor page URL')
    youtube = models.URLField(
        help_text='Your YouTube channel or user account URL')
```

A 'Social media settings' link will appear in the Wagtail admin 'Settings' menu.

2.1 Edit handlers

Settings use edit handlers much like the rest of Wagtail. Add a *panels* setting to your model defining all the edit handlers required:

```
@register_setting
class ImportantPages(BaseSetting):
    donate_page = models.ForeignKey(
        'wagtailcore.Page', null=True, on_delete=models.SET_NULL)
    sign_up_page = models.ForeignKey(
        'wagtailcore.Page', null=True, on_delete=models.SET_NULL)

    panels = [
        PageChooserPanel('donate_page'),
        PageChooserPanel('sign_up_page'),
    ]
```

2.2 Appearance

You can change the label used in the menu by changing the verbose_name of your model.

You can add an icon to the menu by passing an 'icon' argument to the `register_setting` decorator:

```
@register_setting(icon='icon-placeholder')
class SocialMediaSettings(BaseSetting):
    # ...
```

For a list of all available icons, please see the Wagtail style guide.

CHAPTER 3

Using the settings

Settings are designed to be used both in Python code, and in templates.

3.1 Using in Python

If access to a setting is required in the code, the `BaseSetting.for_site` method will retrieve the setting for the supplied site:

```
def view(request):
    social_media_settings = SocialMediaSettings.for_site(request.site)
    ...
```

3.2 Using in templates

Add the `request` and `wagtailsettings` context processors to your settings:

```
from django.conf import global_settings
TEMPLATE_CONTEXT_PROCESSORS = global_settings.TEMPLATE_CONTEXT_PROCESSORS + (
    'django.core.context_processors.request',
    'wagtailsettings.context_processors.settings',
)
```

Then access the settings through `settings`:

```
{{ settings.app_label.SocialMediaSettings.instagram }}
```

If you are not in a RequestContext, then context processors will not have run, and the `settings` variable will not be availble. To get the `settings`, use the provided template tags:

```
{% load wagtailsettings_tags %}
{% get_settings %}
{{ settings.app_label.SocialMediaSettings.instagram }}
```

Note: You can not reliably get the correct settings instance for the current site from this template tag, as the request object is not available. This is only relevant for multisite instances of Wagtail though, so most developers will not have to worry.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search