Wagtail News Documentation

Release 0.16.0

Takeflight

Sep 27, 2017

Contents

Contents:

CHAPTER 1

Setup

`wagtailnews` is compatible with Wagtail 1.6 and higher, Django 1.9 and higher, and Python 3.4 and higher.

Step 1

Install `wagtailnews` using pip:

```
pip install wagtailnews
```

Step 2

Add `wagtailnews` and `wagtail.contrib.wagtailroutablepage` to your `INSTALLED_APPS` in settings:

```
INSTALLED_APPS += [
    'wagtailnews',
    'wagtail.contrib.wagtailroutablepage',
]
```

CHAPTER 2

Usage

To start using Wagtail News, you have to define a news index model, and a news item model.

The news item model stores each news post that you make. The news item model is a regular Django model that inherits from *AbstractNewsItem*. Add any fields that you want to this model:

```
from wagtailnews.models import AbstractNewsItem, AbstractNewsItemRevision
from wagtail.wagtailcore.fields import RichTextField

class NewsItem(AbstractNewsItem):
    title = models.CharField(max_length=100)
    body = RichTextField()

    panels = [
        FieldPanel('title'),
        FieldPanel('body'),
    ]

    def __str__(self):
        return self.title

# This table is used to store revisions of the news items.
class NewsItemRevision(AbstractNewsItemRevision):
    # This is the only field you need to define on this model.
    # It must be a foreign key to your NewsItem model,
    # be named 'newsitem', and have a related_name='revisions'
    newsitem = models.ForeignKey(NewsItem, related_name='revisions')
```

The panels can be customised using the `panels` attribute, or a completely custom edit handler can be used by setting the `edit_handler` attribute. See the Wagtail docs for more information.

The news index model is a subclass of the Wagtail Page model. This page defines where your news page appears on your site. You can define any extra fields you need on this page. This page needs to inherit from `NewsItemMixin`, and be registered using the *wagtailnews.decorators.newsindex()* decorator. The `newsitem_model` attribute defines which news item model to use for this news index.

```
from wagtail.wagtailcore.models import Page
from wagtailnews.decorators import newsindex
from wagtailnews.models import NewsIndexMixin

@newsindex
```

```
class NewsIndex(NewsIndexMixin, Page):
    newsitem_model = 'NewsItem'
```

Make and run migrations, and everything should be waiting for you in the Wagtail admin! First, create a new `NewsIndex` page somewhere in your page tree. Then, click the "News" link in the side bar. From here, you can create and manage news items for this news index.

CHAPTER 3

Forms, widgets, and blocks

Wagtail news integrates with Wagtail edit handlers and stream field blocks.

NewsChooserPanel

class wagtailnews.edit_handlers.**NewsChooserPanel**(*field_name*)

An edit handler for editors to pick a news item. Takes the field name as the only argument. For example:

```
class FooPage(Page):
    news_item = models.ForeignKey('news.NewsItem')

    content_panels = Page.content_panels + [
        NewsChooserPanel('news_item')
    }
```

NewsChooserBlock

class wagtailnews.blocks.**NewsChooserBlock**(*target_model*, ***kwargs*)

A StreamField block for editors to pick a news item. Takes the news item class as its only argument. For example:

```
class FooPage(Page):
    content = StreamField([
        ('text', RichTextField()),
        ('news', NewsChooserBlock('news.NewsItem')),
    ])
```

CHAPTER 4

RSS Feed

wagtailnews supports RSS feeds!

Custom RSS feed fields

wagtailnews support of RSS feeds comes from Django's syndication feed framework. Wagtail News provides a basic implementation, but you will need to customise it to suit your news models. For example, to add a custom <description> for your news items:

```
from wagtailnews.feeds import LatestEntriesFeed

class MyNewsFeed(LatestEntriesFeed):
    def item_description(self, item):
        return item.description
```

Your custom Feed class can then be added to your news index by setting the *feed_class* attribute:

```
@newsindex
class NewsIndex(NewsIndexMixin, Page):
    feed_class = MyNewsFeed
```

Find out more about Feed classes in the Django docs: The syndication feed framework.

Linking to RSS feed

A link to the RSS feed can be created in a template like this:

```
{% load wagtailroutablepage_tags %}
<a href="{% routablepageurl page "feed" %}">RSS</a>
```

The Wagtail docs have more information on the routablepageurl() template tag.

CHAPTER 5

Reference

wagtailnews.decorators.**newsindex**(*cls*)
Register a *NewsIndexMixin* page. News indexes need to be registered before news items can be managed through the admin.

```
@newsindex
class NewsIndex(NewsIndexMixin, Page):
    newsitem_model = NewsItem
```

News items

class wagtailnews.models.**AbstractNewsItem**

Fields

AbstractNewsItem.**date**
A DateTimeField that records the published date of this news item. It is automatically set when the news item is created. You can add it to the AbstractNewsItem.panels list if you want editors to be able to customise the date. If set in the future, the post will not appear on the front end.

AbstractNewsItem.**live**
A BooleanField that indicates if this news item is live. A live news item might have unpublished drafts.

Attributes

AbstractNewsItem.**template**
The template to use for this news item. Defaults to app_label/model_name.html.

Methods

AbstractNewsItem.**get_nice_url**()
Make a slug to put in the URL. News items are fetched using their ID, which is also embedded in the URL, this slug just makes the URLs nicer. The slug does not need to be unique. By default, it is generated from slugify(str(self)).

AbstractNewsItem.**get_template**(*request*)
Get the template for this news item. See also *AbstractNewsItem.template*.

AbstractNewsItem.**get_context**(*request*, **args*, ***kwargs*)
Build a context dictionary for the template. The default implementation gets the context from the news index, and adds the news item as `newsitem`.

AbstractNewsItem.**url_suffix**()
Return the URL of this news item relative to the news index.

```
>>> newsitem.url_suffix()
'2016/08/11/1234-my-news-item/'
```

See also *AbstractNewsItem.get_nice_url()*.

AbstractNewsItem.**url**()
Return the URL of this news item, using the news indexes `url` attribute.

AbstractNewsItem.**full_url**()
Return the URL of this news item, using the news indexes `full_url` attribute.

News index

class wagtailnews.models.**NewsIndexMixin**

Attributes

NewsIndexMixin.**newsitem_model**
The news item model to use for this news index. The news item model must be a subclass of *AbstractNewsItem*. This can either be the name of the model as a string, such as 'NewsItem' or 'myapp.NewsItem', or the actual news item class.

NewsIndexMixin.**feed_class**
The `Feed` class to use to create the RSS feed. See *RSS Feed* for more details.

NewsIndexMixin.**subpage_types**
Defaults to an empty list. News indexes with subpages are not supported.

Methods

classmethod NewsIndexMixin.**get_newsitem_model**()
Get the news item model for this news index. See *NewsIndexMixin.newsitem_model*.

Routes

The functionality of news indexes come from Wagtail's RoutablePageMixin. The following routes are defined:

index The root url which shows all news items.

```
>>> newsindex.reverse_subpage('index')
''
```

year Displays all news items from a year.

```
>>> newsindex.reverse_subpage('year', kwargs={'year': '2016'})
'2016/'
```

month Displays all news items from a month.

```
>>> newsindex.reverse_subpage('month', kwargs={'year': '2016', 'month': '08'})
'2016/08/'
```

day Displays all news items from a day.

```
>>> newsindex.reverse_subpage('day', kwargs={'year': '2016', 'month': '08',
↪'day': '15'})
'2016/08/15/'
```

post Shows a single news item.

```
>>> newsindex.reverse_subpage('post', kwargs={
...     'year': '2016', 'month': '08', 'day': '15',
...     'pk': newsitem.pk, 'slug': newsitem.get_nice_url()})
'2016/08/15/1234-my-news-item/'
```

See also *AbstractNewsItem.get_nice_url()* and *AbstractNewsItem.url_suffix()*.

feed Show the RSS feed.

```
>>> newsindex.reverse_subpage('rss')
'rss/'
```

See also *RSS Feed*.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

w

Index