cmsplugin-blog

Release 1.1.2 post 0

Øyvind Saltvik

March 22, 2016

Contents

cmsplugin-blog is really simple to set up on a working installation of django CMS.

CHAPTER 1

Requirements

- django CMS 2.2
- djangocms-utils 0.9.5+
- simple-translation 0.8.5+
- jQuery 1.4.4+
- jQuery UI 1.8.1+
- django-tagging 0.3+
- django-missing
- django-guardian (optional)

On Django 1.2.7: * django-cbv

Note: jQuery can be provided either by locally or linking to a public server, like Google's or Microsoft's CDN.

1.1 Installation

Install `cmsplugin-blog` from pypi:

```
pip install cmsplugin-blog
```

Note: When installing the cmsplugin-blog using pip django-tagging, django-missing, djangocms-utils, and simple-translation will be installed automatically.

CHAPTER 2

Configuration and setup

2.1 Settings

Add the following apps to your INSTALLED_APPS which enable cmsplugin-blog and required or highly recommended applications/libraries):

- 'cmsplugin_blog', cmsplugin-blog itself
- 'djangocms_utils', utilities and extensions to django CMS
- 'simple_translation', enables multilingual features
- 'tagging', enables tagging of posts
- 'staticfiles', for serving static files
- 'missing', provides improved slug generation
- 'guardian', provides per-object-permissions (see docs for django-guardian)

Add required settings:

```
INSTALLED_APPS = (
    ...
    'cmsplugin_blog',
    'djangocms_utils',
    'simple_translation',
    'tagging',
    'staticfiles',
    'missing',
    'guardian', # optional
    ...
)
```

For Django < 1.3 you need django-cbv for support for class-based views and you should also add cbv.middleware.DeferredRenderingMiddleware to MIDDLEWARE_CLASSES

```
MIDDLEWARE_CLASSES = (
    ...
    'cbv.middleware.DeferredRenderingMiddleware',
)
```

2.1.1 Static content

Set the STATIC_ROOT and STATIC_URL settings for django-staticfiles.:

```
STATIC_ROOT = '/projectpath/static/'
STATIC_URL = '/static/'
```

2.1.2 jQuery and jQuery UI

Add the following settings to add jQuery UI

```
JQUERY_JS = 'https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js'
JQUERY_UI_JS = 'https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.12/jquery-ui.min.js'
JQUERY_UI_CSS = 'http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.12/themes/smoothness/jquery-ui
```

Or download the sources and make them available locally:

```
JQUERY_UI = '/path/to/jquery/'
JQUERY_JS = '%sjs/jquery-1.4.4.min.js' % JQUERY_UI
JQUERY_UI_JS = '%sjs/jquery-ui-1.8.9.custom.min.js' % JQUERY_UI
JQUERY_UI_CSS = '%scss/smoothness/jquery-ui-1.8.9.custom.css' % JQUERY_UI
```

2.1.3 Multilingual blog

If you are interested in multilingual blog, add `cmsplugin_blog.middleware.MultilingualBlogEntriesMiddlewar` to `MIDDLEWARE_CLASSES`

```
MIDDLEWARE_CLASSES = (
    ...
    'cmsplugin_blog.middleware.MultilingualBlogEntriesMiddleware',
)
```

2.1.4 Blog entry placeholders

You can create multiple placeholders for each blog entry. This is useful for creating extra fields like excerpt, images, etc.:

```
CMSPLUGIN_BLOG_PLACEHOLDERS = ('first', 'second', 'third')
```

2.2 Update the database

Next, you need to update the database with the fields required by cmsplugin-blog:

```
python manage.py syncdb

# or if south is installed
python manage.py syncdb --all
python manage.py migrate --fake
```

CHAPTER 3

Templates

In your `template` folder create a template adapted to your site as `cmsplugin_blog/cmsplugin_blog_base.html`.

For example, if you have a template called `base.html` which has a block called `body` create a template that looks like this:

```
{% extends "base.html" %}

{% block body %}
    {% block left-col %}{% endblock %}
    {% block right-col %}{% endblock %}
{% endblock %}
```

Note: The cmsplugin-blog uses the block names `left-col` and `right-col` by default.

CHAPTER 4

Sitemaps

If you use the sitemaps framework in your cms, you can add your blog entry pages to the sitemaps.xml file by including the sitemap class in your urls.py.

e.g.

```
from cms.sitemaps import CMSSitemap
from cmsplugin_blog.sitemaps import BlogSitemap

urlpatterns = patterns('',
    url(r'^sitemap.xml$', 'django.contrib.sitemaps.views.sitemap', {
        'sitemaps': {
            'cmspages': CMSSitemap,
            'blogentries': BlogSitemap
        }
    }),
    url(r'^', include('cms.urls'))
)
```

CHAPTER 5

Creating the blog

To create a blog you need to create a page which will contain the root of the blog. From this page the entire blog is shown.

Create a page in the CMS. Under `Advanced settings` `Application`, select `Blog Apphook`.

Do this for each language you want to show posts in. A restart of the server afterwards is mandatory due to caching.

That should be it! Now you can spend countless hours and nights thinking about what you were supposed to write about that wasn't kittens or other cute furry animals.