django-spotnet Documentation

Release 0.1

Alexander van Ratingen

March 19, 2015

Contents

This documentation covers the 0.1 release of django-spotnet, a simple pluggable application to manage spotnet posts for Django-powered websites.

This app includes all you need to get started with spotnet. It everything handles retrieving posts from your NNTP server, to pushing them to your download server (eg: Sabnzbd+ or NZBGet).

To get up and running quickly, consult the *quick-start guide*, which describes all the necessary steps to install django-spotnet and start using spotnet. For more detailed information read through the documentation listed below.

Contents:

CHAPTER 1

Quick start guide

As this is an app for websites built on the Django framework, you'll need to have that installed. The minimum required versions are Python 2.6 (no python 3 support yet) and Django 1.2.7. To store the posts, you'll need a database. Though any database that's supported by Django ORM is supported by spotnet, you'll probably want to go with something more serious as your post table grows.

1.1 Installing django-spotnet

There are several ways to install django-spotnet:

- Automatically, via a package manager.
- Manually, by downloading a copy of the release package and installing it yourself.
- Manually, by performing a Mercurial/Git checkout of the latest code.

It is also highly recommended that you learn to use virtualenv for development and deployment of Python software; `virtualenv` provides isolated Python environments into which collections of software (e.g., a copy of Django, and the necessary settings and applications for deploying a site) can be installed, without conflicting with other installed software. This makes installation, testing, management and deployment far simpler than traditional site-wide installation of Python packages.

1.1.1 Automatic installation via a package manager

Several automatic package-installation tools are available for Python; the most popular are easy_install and pip. Either can be used to install django-spotnet.

Using `easy_install`, type:

```
easy_install -Z django-spotnet
```

Note that the `-Z` flag is required, to tell `easy_install` not to create a zipped package; zipped packages prevent certain features of Django from working properly.

Using `pip`, type:

```
pip install django-spotnet
```

It is also possible that your operating system distributor provides a packaged version of django-spotnet. Consult your operating system's package list for details, but be aware that third-party distributions may be providing older versions of django-spotnet, and so you should consult the documentation which comes with your operating system's package.

1.1.2 Manual installation from a downloaded package

If you prefer not to use an automated package installer, you can download a copy of django-spotnet and install it manually. The latest release package can be downloaded from django-spotnet's listing on the Python Package Index.

Once you've downloaded the package, unpack it (on most operating systems, simply double-click; alternately, type `tar zxvf django-spotnet-0.1.tar.gz` at a command line on Linux, Mac OS X or other Unix-like systems). This will create the directory `django-spotnet-0.1`, which contains the `setup.py` installation script. From a command line in that directory, type:

```
python setup.py install
```

Note that on some systems you may need to execute this with administrative privileges (e.g., `sudo python setup.py install`).

1.1.3 Manual installation from a Mercurial checkout

If you'd like to try out the latest in-development code, you can obtain it from the django-spotnet repository, which is hosted at Bitbucket and uses Mercurial for version control. To obtain the latest code and documentation, you'll need to have Mercurial installed, at which point you can type:

```
hg clone http://bitbucket.org/Blue/django-spotnet
```

You can also obtain a copy of a particular release of django-spotnet by specifying the `-r` argument to `hg clone`; each release is given a tag of the form `vX.Y`, where "X.Y" is the release number. So, for example, to check out a copy of the 0.1 release, type:

```
hg clone -r v0.1 http://bitbucket.org/Blue/django-spotnet
```

In either case, this will create a copy of the django-spotnet Mercurial repository on your computer; you can then add the `django-spotnet` directory inside the checkout your Python import path, or use the `setup.py` script to install as a package.

1.2 Basic configuration and use

Once installed, you can add django-spotnet to any Django-based project you're developing. Adding `spotnet` to the `INSTALLED_APPS` setting of your project, and set the required setting `SPOTNET_SERVER_HOST` to the hostname of your NNTP server.

For more control, there are a number of additional settings to customize things. There are a few of these that you might want to override.

`SPOTNET_SERVER_PORT` The port on the NNTP server to connect to.

`SPOTNET_SERVER_USERNAME` The username for connecting to the NNTP server.

`SPOTNET_SERVER_PASSWORD` The password for connecting to the NNTP server. You should make sure this is kept secret, so you might want to keep it from your VCS history.

`SPOTNET_SERVER_SSL` Set this to `True` if you want to encrypt connections to the NNTP server.

`SPOTNET_CLEANUP_MINAGE` The minimum age of a post, in days, to get deleted from the database. Posts older than this are deleted when you update the database.

`SPOTNET_POST_PER_PAGE` The number of posts to show in the paginators when viewing the post list.

`SPOTNET_ALLOW_NZB_UPLOAD` Allow users to upload nzb files directly to downloadservers, thus bypassing the spotnet posts alltogether.

SPOTNET_ANONYMOUS_ACTION What to do when an anonymous visitor views the spotnet pages. You can set this to either 'allow', '404', '403' or 'login', for respectively granting them full access, responding with a 404 error, responding with a 403 error, or redirecting them to the login page.

SPOTNET_DOWNLOAD_SERVERS A dictionary of download servers, that is the servers that download the posts and nzbs. View the documentation on this to learn *how to define download servers*.

1.2.1 Setting up URLs

The app includes a Django `URLconf` which sets up URL patterns for *the views in django-spotnet*. This `URLconf` can be found at `spotnet.urls`, and so can simply be included in your project's root URL configuration. For example, to place the URLs under the prefix `/spotnet/`, you could add the following to your project's root `URLconf`:

```
(r'^spotnet/', include('spotnet.urls')),
```

This would then be the index page for using spotnet. To completely customize the url locations, add the urlpatterns for the *included views* yourself. If you go down this road, do make sure to use the same url names as the default urlconf.

1.2.2 Templates

By default, spotnet uses very simple templates for it's views. You probably want to override these by creating your own templates. These should be findable under names like `spotnet/list.html`. See the included templates and their views for the names and available context variables.

Note that all of these templates are rendered using a `RequestContext` and so will receive any additional variables provided by context processors.

CHAPTER 2

Releases

2.1 spotnet 0.1

The initial release

CHAPTER 3

Download servers

To define your own download servers, ...

CHAPTER 4

Views

These are all the views included in this app.

`spotnet.views.`**`search`**(*request*[, *search=None*[, *cats=None*[, *scats=None*]]])
    Show a list of posts. Filter by passing the optional arguments.

`spotnet.views.`**`viewpost`**(*request*, *id*)
    Show a single post.

`spotnet.views.`**`download`**(*request*, *id*[, *dls=None*])
    Download a post using a download server. Pass a downloadserver name, or it will the default download server.

`spotnet.views.`**`downloaded`**(*request*)
    Show the list of all downloaded posts.

`spotnet.views.`**`download_nzb`**(*request*, *id*)
    Download the nzb file for a spot.

`spotnet.views.`**`update`**(*request*)
    Start updating the spot database. This only works if you set the optional setting `SPOTNET_UPDATE_ALLOW_INPAGE` to `True`.

4.1 Helpers

`spotnet.views.`**`view_related_post_list`**(*request*, *objects*, *page*, *title*[, *extra_actions={}*])
    This is a helper function that other views can call to show a list of model instances that have a foreign key to a spotnet post.

    The view `downloaded` uses this helper. View the source for details on how to use this.

Parameters

- **request** – the `HttpRequest` object
- **objects** – the queryset for a model that has a `ForeignKey` field to the spotnet `Post` model called `post`
- **page** – the page number in the paginator
- **title** – the title string for this page
- **extra_actions** – a dict mapping action names to `Action` instances that can be applied to the related objects

Frequently asked questions

These are some of the questions I've gotten about this project.

`This is pretty awesome` Ok, this is not really a question, but let me give it a go. Yes, it most certainly is!

See also:

- The source code, to find out about django-spotnet internals.

Python Module Index