
django-periodicals Documentation

Release 0.5.0

Steve Schwarz

December 14, 2013

Contents

1	django-periodicals	3
1.1	Documentation	3
1.2	Quickstart	3
1.3	Features	4
2	Installation	7
2.1	Installing from PyPi	7
2.2	Installing from GitHub	7
2.3	Install a Haystack Backend	7
3	Usage	9
3.1	settings.py	9
3.2	urls.py	10
3.3	Database Setup	10
3.4	Disabling Adding/Displaying Links	10
4	Try the Demo Project!	13
5	Contributing	15
5.1	Types of Contributions	15
5.2	Get Started!	16
5.3	Pull Request Guidelines	16
5.4	Tips	17
6	Credits	19
6.1	Development Lead	19
6.2	Contributors	19
7	History	21
7.1	0.5.0 (2013-10-06)	21

Contents:

django-periodicals

A Django application for periodical/magazine websites with fully cross linked indices on Periodical, Issue, Article, Author, Article Series and Tags. Provides full text search of article titles and descriptions. A complete set of templates are provided. A sitemap is also dynamically generated.

1.1 Documentation

The full documentation is at <http://django-periodicals.rtd.org>.

1.2 Quickstart

Simple to install from a package using `pip` which will install all it's dependencies (coming soon). For now see [installing-from-github](#).

```
pip install django-periodicals
```

Install two packages manually to get newer versions than are currently in PyPi:

```
$ pip install -e git://github.com/saschwarz/django-recaptcha.git#egg=django-recaptcha
```

```
$ pip install -e git://github.com/nemith/django-tagging.git@dev-django1.5#egg=django_tagging-dev
```

Install a search backend for use by [Haystack](#). To start install Whoosh:

```
$ pip install Whoosh
```

1.2.1 settings.py

Add `periodicals` and the other applications it uses to `INSTALLED_APPS`:

```
INSTALLED_APPS = (  
    ...  
    'haystack',  
    'tagging',  
    'captcha',  
)
```

```
    'periodicals',  
)
```

Configure your Haystack backend. Here is an example using `Whoosh`:

```
HAYSTACK_CONNECTIONS = {  
    'default': {  
        'ENGINE': 'haystack.backends.whoosh_backend.WhooshEngine',  
        'PATH': os.path.join(os.path.abspath(os.path.dirname(__file__)), 'whoosh_periodicals_index'),  
        'STORAGE': 'file',  
        'POST_LIMIT': 128 * 1024 * 1024,  
        'INCLUDE_SPELLING': True,  
        'BATCH_SIZE': 100,  
    },  
}
```

Configure your reCAPTCHA keys - only used when users add links to Articles or Issues:

```
RECAPTCHA_PRIVATE_KEY = "your-recaptcha-private-key"  
RECAPTCHA_PUBLIC_KEY = "your-recaptcha-public-key"
```

1.2.2 urls.py

Choose a URL prefix at which to base the application:

```
...  
import periodicals  
  
urlpatterns = patterns('',  
    ...  
    url(r'^admin/', include(admin.site.urls)),  
    url(r'^periodicals/', include(periodicals.urls)),  
)
```

1.2.3 Management Commands

```
$ python manage.py syncdb
```

1.3 Features

I developed `django-periodicals` to provide a searchable index for a printed/online magazine. I wanted all the meta data to be fully cross linked. So users can easily browse all articles for an author, all articles in an issue, all articles in a series/category, all articles tagged with a keyword and so forth.

I turned it in to a standalone application when I ported it to Django 1.5. Here are the features:

- Provides Django models for Periodicals, Issues, Articles, Authors, Tags and Links to external material.
- A full set of templates are provided including:
 - Individual Periodical pages with yearly indices.
 - Fully cross-linked indexes of Authors, Issues, Article Series, Tags, and Articles.
 - Search across Article titles and descriptions.

- Tagging:
 - * Per article.
 - * Index pages per tag.
 - * Tag cloud.
- Moderated user added links of blog posts and other web resources to each Issue and Article. Spam protection by [reCAPTCHA](#) and requiring approval by the admin. This feature can be disabled via setting.
- Django admin forms for data entry.
- Sitemap support.
- Support for Python 2.6, 2.7 and Django 1.5 and 1.6.
- Travis CI unit tests.
- See [django-periodicals in action at Googility](#).

Installation

You can install from [PyPi](#) and manually add some packages or install from a GitHub checkout and use `pip` and a `requirements.txt` file.

2.1 Installing from PyPi

Simple to install from a package using `pip` which will install most of its dependencies:

```
$ pip install django-periodicals
```

Install two packages/applications manually to get newer versions than are currently in PyPi:

```
$ pip install -e git://github.com/saschwarz/django-recaptcha.git#egg=django-recaptcha
```

```
$ pip install -e git://github.com/nemith/django-tagging.git@dev-django1.5#egg=django_tagging-dev
```

Continue installing with [Install a Haystack Backend](#) below.

2.2 Installing from GitHub

This lets you install all the requirements using `pip` and the `requirements.txt` file:

```
$ git clone https://github.com/saschwarz/django-periodicals.git
```

```
$ cd django-periodicals
```

```
$ pip install -r requirements.txt
```

```
$ python setup.py install
```

2.3 Install a Haystack Backend

Install a search backend for use by [Haystack](#). To start install Whoosh:

```
$ pip install Whoosh
```

Try the Demo Project!

Usage

`django-periodicals` integrates a couple other applications to provide it's features.

3.1 settings.py

Add `periodicals` and the other applications it uses to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    'haystack',
    'tagging',
    'captcha',
    'periodicals',
)
```

Configure your Haystack backend. Here is an example using `Whoosh`:

```
HAYSTACK_CONNECTIONS = {
    'default': {
        'ENGINE': 'haystack.backends.whoosh_backend.WhooshEngine',
        'PATH': os.path.join(os.path.abspath(os.path.dirname(__file__)), 'whoosh_periodicals_index'),
        'STORAGE': 'file',
        'POST_LIMIT': 128 * 1024 * 1024,
        'INCLUDE_SPELLING': True,
        'BATCH_SIZE': 100,
    },
}
```

Configure your reCAPTCHA keys - only used when users add links to Articles or Issues (this feature can be disabled):

```
RECAPTCHA_PRIVATE_KEY = "your-recaptcha-private-key"
RECAPTCHA_PUBLIC_KEY = "your-recaptcha-public-key"
```

Configure you Site in the Django Site application - only used when users add links to Articles or Issues *and* when you haven't disabled email notifications.

3.2 urls.py

Choose a URL prefix at which to base the application:

```
...
import periodicals

urlpatterns = patterns('',
    ...
    url(r'^admin/', include(admin.site.urls)),
    url(r'^periodicals/', include(periodicals.urls)),
)
```

3.3 Database Setup

```
$ python manage.py syncdb
```

3.3.1 Optional Settings

You can control the display format for Author, Periodical, and Issue instances and their URL slugs through the following `settings.py` values. The default values are shown below:

```
PERIODICALS_AUTHOR_FORMAT = "%(last_name)s, %(first_name)s %(middle_name)s %(postnomial)s"
PERIODICALS_AUTHOR_SLUG_FORMAT = "%(last_name)s %(first_name)s %(middle_name)s %(postnomial)s"

PERIODICALS_PERIODICAL_FORMAT = "%(name)s"
PERIODICALS_PERIODICAL_SLUG_FORMAT = "%(name)s"

PERIODICALS_ISSUE_FORMAT = "Vol. %(volume)s No. %(issue)s"
PERIODICALS_ISSUE_SLUG_FORMAT = "%(volume)s %(issue)s"
```

3.4 Disabling Adding/Displaying Links

By default visitors can add moderated links to each Issue or Article. Once approved via the admin they are displayed on the appropriate Issue/Article page. To disable this feature and the sections within pages displaying links add this to `settings.py`:

```
PERIODICALS_LINKS_ENABLED = False
```

By default when links are added an email is sent to managers configured for the Site in the Django admin. To disable this feature add this to `settings.py`:

```
PERIODICALS_EMAIL_NOTIFY = False
```

3.4.1 Entering Data

Use the Django admin pages for the Periodical application to enter data. It is easiest to proceed in this order:

1. Create a Periodical.
2. Create an Issue and select the created Periodical.

Try the Demo Project!

Check out `django-periodicals` as described in *Installing from GitHub*. There is a directory called `demo` in the checkout directory that contains a fully functional installation.

It is designed to show how to configured your project to use this application.

I also overrode the `base.html` template and added in some trivial [Twitter Bootstrap](#) styling so the pages wouldn't be unstyled. `django-periodicals` does not require the use of Bootstrap.

Follow these steps:

```
$ cd django-periodicals
$ virtualenv demoenv
$ source demoenv/bin/activate
$ pip install -r requirements-test.txt
$ pip install Django
$ cd demo
$ python manage.py syncdb
$ python manage.py loaddata demo_tagging.json
$ python manage.py loaddata demo_periodicals.json
$ python manage.py update_index
$ python manage.py runserver
```

So now you can visit the demo site at <http://127.0.0.1:8000/>.

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at <https://github.com/saschwarz/django-periodicals/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

5.1.4 Write Documentation

django-periodicals could always use more documentation, whether as part of the official django-periodicals docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/saschwarz/django-periodicals/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *django-periodicals* for local development.

1. Fork the *django-periodicals* repo on GitHub.

2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-periodicals.git
```

3. Install your local copy into a virtualenv. Assuming you have `virtualenvwrapper` installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-periodicals
$ cd django-periodicals/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass `flake8` and the tests, including testing other Python versions with `tox`:

```
$ flake8 periodicals tests
$ python setup.py test
$ tox
```

To get `flake8` and `tox`, just `pip` install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/saschwarz/django-periodicals/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_periodicals
```

Credits

6.1 Development Lead

- Steve Schwarz <steve@agilitynerd.com>

6.2 Contributors

None yet. Why not be the first?

History

7.1 0.5.0 (2013-10-06)

- Migrated to Django 1.6 using class based views.