Django Wordpress API Documentation

Release 0.1.0

Swapps

Jun 28, 2017

Contents

Contents:

CHAPTER 1

Django Wordpress API

Easily Install your Wordpress blog in your Django project

This package allows to communicate easily with any wordpress project that is using WP REST API v1 .

Even though the WP REST API package is already on the 2 version; it is still on beta so it was decided that this package will only support v1 until v2 is out of beta.

Documentation

The full documentation is at https://django-wordpress-api.readthedocs.org.

Quickstart

Install Django Wordpres API:

```
pip install django-wordpress-api
```

Then use it in a project:

```
import wordpress_api
```

Features

- Connect to an external wordpress application
- Retrieves all the blog posts ordered by pages
- Filter blog posts using several of the available filters in WP REST API
- Search blog posts using a keyword

- order the blog posts by several attributes like author, title, type, etc; ascending and descending order (default order is descending date)
- Retrieve posts with a different type than "post"
- Four Views to display the blog page, The Post detail, The Posts filtered by category and the Posts filtered by tag; All of this with the search by keyword option

Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install -r requirements.txt
(myenv) $ pip install -r requirements_test.txt
(myenv) $ python manage.py test
```

Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-wordpress-api
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-wordpress-api
$ pip install django-wordpress-api
```

CHAPTER 3

Integration

This section describes step by step integration of django_wordpress_api with your application.

Configure django-wordpress_api inside your aplication

Add this app to your `INSTALLED_APPS` in your settings file:

```
INSTALLED_APPS += ('wordpress_api',)
```

You need two settings variables to be able to use the package:

```
WP_URL = http://your-wordpress-app.com/
BLOG_POSTS_PER_PAGE = <number-of-blogs-to-display-per-page>
```

Remember to add WP REST API v1 to http://your-wordpress-app.com/ or this package will be useless.

Add django-wordpress-api

Add django-wordpress-api urls to your URL general configuration:

```
url(r'^blog/', include('wordpress_api.urls')),
```

Multilingual support

At version 0.1.8 multilingual support was added. To use it, you need to install WPML and wpml wp rest api adapter plugin by aaltomeri inside your wordpress site and set the following variable inside your settings.

```
WP_API_ALLOW_LANGUAGE = True
```

Inside the views, the language is supported using `django.utils.translation.get_language`. If you are not using django translation, you can use the WPApiConnector.get_posts method directly and pass the language as the lang parameter. You can check how this work at `wordpress_api/utils.py`

Page cache

At version 0.1.18 cache support was added to all django wordpress api related pages. To activate it, just set the following setting.

```
WP_API_BLOG_CACHE_TIMEOUT = 60 * 60 * 24
```

RSS Feed

At version 0.1.23 a RSS Feed was added. You may use it importing LatestEntriesFeed from wordpress_api.feed_views and adding it to your urls configuration.

```
url(r'^feed/$', LatestEntriesFeed()),
```

If you want to modify the title or the description, just create your own class and inherit LatestEntriesFeed.

CHAPTER 4

Usage

To use Django Wordpress API in a project:

```
import wordpress_api
```

The django-wordpress-api has two main features: the WPApiConnector and the Views that uses it. If you want to use the pre defined views, just add wordpress-api-urls to your project.

The basic django-wordpress-api urls are:

```
http://localhost:8000/blog/; display the blog list

http://localhost:8000/(?P<slug>[-\w]+)/; displays the detail of a blog identified␣
↪with the given slug

http://localhost:8000/category/(?P<slug>[-\w]+)/; displays the blogs in the category␣
↪identified with the given slug

http://localhost:8000/tag/(?P<slug>[-\w]+)/; displays the blogs in the tag identified␣
↪with the given slug

http://localhost:8000/author/(?P<slug>[-\w]+)/; displays the blogs written by the␣
↪author identified with the slug
```

Else, If you want to retrieve the blog posts in your custom views, you can use directly the WPApiConnector and its methods. You can check them at `wordpress_api/utils.py`

CHAPTER 5

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/swappsco/django-wordpress-api/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/swappsco/django-wordpress-api/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 :)

Get Started!

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

1. Fork the *django-wordpress-api* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/django-wordpress-api.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-wordpress-api
   $ cd django-wordpress-api/
   $ 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 wordpress_api tests
   $ python manage.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.

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/swappsco/django-wordpress-api/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

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

CHAPTER 6

Credits

Development Lead

- Swapps <dev@swapps.io>

Contributors

None yet. Why not be the first?

CHAPTER 7

History

0.1.0 (2016-09-02)

- First release on PyPI.