mezzanine-buffer Documentation

Release 0.1.0

Alex Tsai

May 03, 2015

Contents:

CHAPTER 1

mezzanine-buffer

Buffer integration for Mezzanine CMS

1.1 Documentation

The full documentation is at https://mezzanine-buffer.readthedocs.org.

1.2 Quickstart

This assumes you already have a Mezzanine install.

Install mezzanine-buffer:

```
pip install mezzanine-buffer  --process-dependency-links
```

Unfortunately, the process-dependency-links is required until buffer-python is updated on pypi.

Then use it in a project:

- Add the following to your installed_apps:

  ```
  "mezzanine_buffer"
  ```

- Create a Buffer account (if you don't have one already)
- Create a Buffer App for your Mezzanine site. You will receive an email with your client key, client secret, and access token
- Enter your client key, client secret, and access token into your Mezzanine site settings.

1.3 Features

- Adds a list of your Buffer profiles to the status section of any *Displayable* admin.
- If the publish_date of *Displayable* is in the future, it will be scheduled for that time.

1.4 TODO

- tests
- proper multi-profile support (buffpy doesn't support it)
- error handling (max 10 updates per profile, rate limits etc)

CHAPTER 2

Installation

At the command line:

```
$ easy_install mezzanine-buffer
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv mezzanine-buffer
$ pip install mezzanine-buffer
```

CHAPTER 3

Usage

To use mezzanine-buffer in a project:

```
import mezzanine-buffer
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/caffodian/mezzanine-buffer/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/caffodian/mezzanine-buffer/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 :)

4.2 Get Started!

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

1. Fork the *mezzanine-buffer* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/mezzanine-buffer.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 mezzanine-buffer
   $ cd mezzanine-buffer/
   $ 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 mezzanine_buffer 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.

4.3 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/caffodian/mezzanine-buffer/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Alex Tsai <caffodian@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2015-05-03)

- First release on PyPI.