django-static-push

Release 0.1.0

December 31, 2016

Contents

Overview

docs	
tests	
package	

Middleware and templatetag for Django to utilize HTTP/2 push for assets included in a Django template. The middleware injects a *Link* header in each response if there are files to be pushed to the client. All files in the template which are suitable for HTTP/2 push should be included with the `staticpush` templatetag instead of the vanilla `static` templatetag. The former simply augments the later and registers the resulting static URL with the middleware.

This package currently supports Apache2 webservers with `mod_http2` enabled, as the actual HTTP/2 push is offloaded to the webserver.

Warning: This is ALPHA code. Do not use in production! It only serves as a proof-of-concept for now. Conditional HTTP/2 push is not supported yet. This means that your site will actually perform worse than over HTTP/1.1 because each response will trigger a push of all incldued assets, irrespective of any cache on the webbrowser.

1.1 Installation

```
pip install django-static-push
```

1.2 Documentation

https://django-static-push.readthedocs.io/en/latest/

1.3 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER 2

Installation

At the command line:

```
pip install django-static-push
```

Usage

To use django-static-push in a project where you want push assets over HTTP/2, add the `StaticPush` middleware to your `settings.py` file and include the django.template.context_processors.request context processor in your templating configuration:

```
MIDDLEWARE = [
    ...,
    'django_static_push.middleware.StaticPush',
]

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                ...,
                'django.template.context_processors.request',
                ...,
            ],
        },
    },
]
```

Now you can use the `staticpush` templatetag in your Django templates:

```
{% load staticpush %}
<link rel="stylesheet" href="{% staticpush 'some/file.css' %}"
```

Make sure that Apache2 mod_http2 has been configured correctly for your webserver:

```
<VirtualHost *:443>
    ...
    Protocols h2 http/1.1
    H2Push on
    ...
</VirtualHost>
```

Each HTTP response will now carry a `Link` header as described in the H2Push documentation, causing Apache2 to send all files included by the `staticpush` templatetag to the webbrowser.

CHAPTER 4

Reference

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When 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.

5.2 Documentation improvements

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

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/fladi/django-static-push/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 code contributions are welcome :)

5.4 Development

To set up *django-static-push* for local development:

1. Fork django-static-push (look for the “Fork” button).
2. Clone your fork locally:

```
    git clone git@github.com:your_name_here/django-static-push.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
    tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`) [1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.

It will be slower though ...

CHAPTER 6

Authors

- Michael Fladischer - https://openservices.at

CHAPTER 7

Changelog

7.1 0.1.0 (2016-01-29)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search