django-github-webhook

Release 0.1.1

May 23, 2017

Contents

CHAPTER 1

Overview

docs	
tests	
package	

A class based view for Django that can act as an receiver for GitHub webhooks. It is designed to validate all requests through their `X-Hub-Signature` headers.

Handling of GitHub events is done by implementing a class method with the same name as the event, e.g. `ping`, `push` or `fork`. See the documentation for more in-depth information and examples.

- Free software: BSD license

Installation

```
pip install django-github-webhook
```

Documentation

https://django-github-webhook.readthedocs.org/

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-github-webhook
```

CHAPTER 3

Usage

To use django-github-webhook in a project where you want to receive webhooks for `push` events:

```
from django_github_webhook.views import WebHookView


class MyWebHookReceiverView(WebHookView):
    secret = 'foobar'

    def push(self, payload, request):
        ''' Do something with the payload and return a JSON serializeable value. '''
        return {'status': 'received'}
```

If the secret has to be dynamically fetched for each request you should override the `get_secret` method:

```
from .models import Hook

class MyWebHookReceiverView(WebHookView):

    def get_secret(self):
        hook = Hook.objects.get(pk=self.request.kwargs['id'])
        return hook.secret
```

Each webhook can receive multiple GitHub events by implementing methods with the same name as the events. Right now the following events are accepted:

- commit_comment
- create
- delete
- deployment
- deployment_status
- fork

- gollum
- issue_comment
- issues
- member
- membership
- page_build
- ping
- public
- pull_request
- pull_request_review_comment
- push
- release
- repository
- status
- team_add
- watch

So in order to accept events of type `fork` and `watch` implement methods as follows. The `payload` parameter gets the already decoded JSON payload from the request body:

```
class MyWebHookReceiverView(WebHookView):

    def fork(self, payload, request):
        print('Forked by {payload[forkee][full_name]}'.format(payload=payload))
        return {'status': 'forked'}

    def watch(self, payload, request):
        print('Watched by {payload[sender][login]}'.format(payload=payload))
```

CHAPTER 4

Reference

CHAPTER 5

Contributing

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

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.

Documentation improvements

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

Feature requests and feedback

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

Development

To set up *django-github-webhook* for local development:

1. Fork django-github-webhook (look for the "Fork" button).
2. Clone your fork locally:

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

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`.

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

0.1.0 (2016-01-29)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search