dj-webhooks Documentation

Release 0.2.2

Daniel Greenfeld

December 19, 2016

Contents

Contents:

CHAPTER 1

dj-webhooks

Django + Webhooks Made Easy

The full documentation is at https://dj-webhooks.readthedocs.org.

1.1 Requirements

- Python 2.7.x or 3.3.2 or higher
- django>=1.5.5
- django-jsonfield>=0.9.12
- django-model-utils>=2.0.2
- django-rq>=0.6.1
- webhooks>=0.3.1

1.2 Quickstart

Install dj-webhooks:

```
pip install dj-webhooks
```

Configure some webhook events:

```
# settings.py
WEBHOOK_EVENTS = (
    "purchase.paid",
    "purchase.refunded",
    "purchase.fulfilled"
)
```

Add some webhook targets:

```
from django.contrib.auth import get_user_model
User = get_user_model()
user = User.objects.get(username="pydanny")

from webhooks.models import Webhook
WebhookTarget.objects.create(
```

```
    owner=user,
    event="purchase.paid",
    target_url="https://mystorefront.com/webhooks/",
    identifier="User or system defined string",
    header_content_type=Webhook.CONTENT_TYPE_JSON,
)
```

Then use it in a project:

```
from django.contrib.auth import get_user_model
User = get_user_model()
user = User.objects.get(username="pydanny")

from djwebhooks.decorators import hook

from myproject.models import Purchase

# Event argument helps identify the webhook target
@hook(event="purchase.paid")
def send_purchase_confirmation(purchase, owner, identifier):
    return {
        "order_num": purchase.order_num,
        "date": purchase.confirm_date,
        "line_items": [x.sku for x in purchase.lineitem_set.filter(inventory__gt=0)]
    }

for purchase in Purchase.objects.filter(status="paid"):
    send_purchase_confirmation(
        purchase=purchase,
        owner=user,
        identifier="User or system defined string"
    )
```

1.3 Storing Redis delivery logs

Note: The only difference between this and the previous example is the use of the redislog_hook.

```
from django.contrib.auth import get_user_model
User = get_user_model()
user = User.objects.get(username="pydanny")

from djwebhooks.decorators import redislog_hook

from myproject.models import Purchase

# Event argument helps identify the webhook target
@redislog_hook(event="purchase.paid")
def send_purchase_confirmation(purchase, owner, identifier):
    return {
        "order_num": purchase.order_num,
        "date": purchase.confirm_date,
        "line_items": [x.sku for x in purchase.lineitem_set.filter(inventory__gt=0)]
    }

for purchase in Purchase.objects.filter(status="paid"):
    send_purchase_confirmation(
```

```
        purchase=purchase,
        owner=user,
        identifier="User or system defined string"
    )
```

1.4 In a queue using django-rq

Warning: In practice I've found it's much more realistic to use the ORM or Redislib webhooks and define seperate asynchronous jobs then to rely on the `djwebhooks.redisq_hook decorator`. Therefore, this functionality is deprecated.

1.5 Features

- Synchronous webhooks
- Delivery tracking via Django ORM.
- Options for asynchronous webhooks.

1.6 Planned Features

- Delivery tracking via Redis and other write-fast datastores.

CHAPTER 2

Installation

At the command line:

```
$ easy_install dj-webhooks
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv dj-webhooks
$ pip install dj-webhooks
```

CHAPTER 3

Usage

To use dj-webhooks in a project:

```
import dj-webhooks
```

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/pydanny/dj-webhooks/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

dj-webhooks could always use more documentation, whether as part of the official dj-webhooks 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/pydanny/dj-webhooks/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 *dj-webhooks* for local development.

1. Fork the *dj-webhooks* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dj-webhooks.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 dj-webhooks
$ cd dj-webhooks/
$ 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 djwebhooks 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/pydanny/dj-webhooks/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_djwebhooks
```

CHAPTER 5

Credits

5.1 Development Lead

- Daniel Greenfeld <pydanny@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.2.2 (2014-05-22)

- Added redislog_hook. This synchronous hook saves the hook results to redis lists.
- Added identifier field to WebhookTarget
- Added identifier argument to orm and redisq senders.
- Added South migrations for Django=<1.6.
- Declared coding in all Python modules.
- Added verbose names to models

6.2 0.2.1 (2014-05-17)

- Removed conf.py file as it just added abstraction.
- Created exlicitly importable hooks. Makes settings management easier.
- Removed utils.py since we no longer do fancy dynamic imports (see previous bullet).
- Coverage now at 100%

6.3 0.2.0 (2014-05-15)

- Refactored the senders to be very extendable.
- Added an ORM based sender.
- Added a redis based sender that uses django-rq.
- Added a *redis-hook* decorator.
- Added admin views.
- Ramped up test coverage to 89%.
- setup.py now includes all dependencies.

6.4 0.1.0 (2014-05-12)

- First release on PyPI.