flask$_{p}ushjackDocumentation$

Release 1.0.0

Derrick Gilland

September 27, 2017

Contents

Flask extension for push notifications on APNS (iOS) and GCM (Android).

CHAPTER 1

Links

- Project: https://github.com/dgilland/flask-pushjack
- Documentation: http://flask-pushjack.readthedocs.org
- PyPi: https://pypi.python.org/pypi/flask-pushjack/
- TravisCI: https://travis-ci.org/dgilland/flask-pushjack

CHAPTER 2

Quickstart

Whether using `APNS` or `GCM`, Flask-Pushjack provides an API client for each.

APNS

```
from flask import Flask
from flask_pushjack import FlaskAPNS

config = {
    'APNS_CERTIFICATE': '<path/to/certificate.pem>'
}

app = Flask(__name__)
app.config.update(config)

client = FlaskAPNS()
client.init_app(app)

with app.app_context():
    token = '<device token>'

    # Send to single device.
    res = client.send(token, alert, **options)

    # List of all tokens sent.
    res.tokens

    # List of any subclassed APNSServerError objects.
    res.errors

    # Dict mapping token => APNSServerError.
    res.token_errors

    # Send to multiple devices.
    client.send([token], alert, **options)

    # Get expired tokens.
    expired_tokens = client.get_expired_tokens()
```

GCM

```
from flask import Flask
from flask_pushjack import FlaskGCM

config = {
    'GCM_API_KEY': '<api-key>'
}

app = Flask(__name__)
app.config.update(config)

client = FlaskGCM()
client.init_app(app)

with app.app_context():
    token = '<device token>'

    # Send to single device.
    res = client.send(token, alert, **options)

    # List of requests.Response objects from GCM Server.
    res.responses

    # List of messages sent.
    res.messages

    # List of registration ids sent.
    res.registration_ids

    # List of server response data from GCM.
    res.data

    # List of successful registration ids.
    res.successes

    # List of failed registration ids.
    res.failures

    # List of exceptions.
    res.errors

    # List of canonical ids (registration ids that have changed).
    res.canonical_ids


    # Send to multiple devices.
    client.send([token], alert, **options)
```

For more details, please see the documentation for pushjack at http://pushjack.readthedocs.org.

Configuration

APNS

`APNS_CERTIFICATE`	File path to certificate PEM file (**must be set**). Default: `None`
`APNS_ENABLED`	Whether to enable sending. Default `True`
`APNS_SANDBOX`	Whether to use sandbox server. Default: `False`
`APNS_DEFAULT_ERROR_TIMEOUT`	Timeout when polling APNS for error after sending. Default: `10`
`APNS_DEFAULT_EXPIRATION_OFFSET`	Message expiration (secs) from now. Default: `2592000` (1 month)
`APNS_DEFAULT_BATCH_SIZE`	Number of notifications to group together when sending.

GCM

`GCM_API_KEY`	API key (**must be set**). Default: `None`
`GCM_ENABLED`	Whether to enable sending. Default `True`

CHAPTER 3

Guide

Installation

Flask-Pushjack requires Python >= 2.6 or >= 3.3.

To install from PyPi:

```
pip install Flask-Pushjack
```

API Reference

The APNS and GCM Flask clients are thin wrappers around the pushjack APNS and GCM clients. For further details, see the pushjack documentation.

APNS

class flask_pushjack.**FlaskAPNS**(*app=None*)
Flask extension for APNS client.

client
Return push notification client associated with current app.

enabled
Return whether client is enabled.

get_expired_tokens(**args*, ***kargs*)
Return expired tokens.

init_app(*app*)
Initialize extension with application configuration.

send(**args*, ***kargs*)
Send push notification to single or multiple recipients.

GCM

class flask_pushjack.**FlaskGCM**(*app=None*)
Flask extension for GCM client.

client
Return push notification client associated with current app.

enabled
Return whether client is enabled.

init_app(*app*)
Initialize extension with application configuration.

send(**args*, ***kargs*)
Send push notification to single or multiple recipients.

CHAPTER 4

Project Info

License

The MIT License (MIT)

Copyright (c) 2015 Derrick Gilland

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Versioning

This project follows Semantic Versioning with the following caveats:

- Only the public API (i.e. the objects imported into the `Flask-Pushjack` module) will maintain backwards compatibility between MINOR version bumps.
- Objects within any other parts of the library are not guaranteed to not break between MINOR version bumps.

With that in mind, it is recommended to only use or import objects from the main module, `Flask-Pushjack`.

Changelog

v1.0.0 (2015-04-28)

- Add `APNS_DEFAULT_BATCH_SIZE=100` config option.
- Pin `pushjack` dependency version to `>=1.0.0`. (**breaking change**)

- Remove `send_bulk` method as bulk sending is now accomplished by the `send` function. (**breaking change**)
- Remove `APNS_HOST`, `APNS_PORT`, `APNS_FEEDBACK_HOST`, and `APNS_FEEDBACK_PORT` config options. These are now determined by whether `APNS_SANDBOX` is `True` or not.
- Remove `APNS_MAX_NOTIFICATION_SIZE` as config option.
- Remove `GCM_MAX_RECIPIENTS` as config option.
- Rename `APNS_ERROR_TIMEOUT` config option to `APNS_DEFAULT_ERROR_TIMEOUT`. (**breaking change**)

v0.1.1 (2015-04-14)

- Pin `pushjack` dependency version to `>=0.1.0, <0.3.0` due to an breaking changes in `pushjack`.

v0.1.0 (2015-03-26)

- First release.

Authors

Lead

- Derrick Gilland, dgilland@gmail.com, dgilland@github

Contributors

How to Contribute

Overview

1. Fork the repo.
2. Build development environment run tests to ensure a clean, working slate.
3. Improve/fix the code.
4. Add test cases if new functionality introduced or bug fixed (100% test coverage).
5. Ensure tests pass.
6. Add yourself to `AUTHORS.rst`.
7. Push to your fork and submit a pull request to the `develop` branch.

Guidelines

Some simple guidelines to follow when contributing code:

- Adhere to PEP8.
- Clean, well documented code.
- All tests must pass.
- 100% test coverage.

Branching

There are two main development branches: `master` and `develop`. `master` represents the currently released version while `develop` is the latest development work. When submitting a pull request, be sure to submit to `develop`. The originating branch you submit from can be any name though.

Continuous Integration

Integration testing is provided by Travis-CI at https://travis-ci.org/dgilland/pydash.

Test coverage reporting is provided by Coveralls at https://coveralls.io/r/dgilland/pydash.

Project CLI

Some useful CLI commands when working on the project are below. **NOTE:** All commands are run from the root of the project and require `make`.

make build

Run the `clean` and `install` commands.

```
make build
```

make install

Install Python dependencies into virtualenv located at `env/`.

```
make install
```

make clean

Remove build/test related temporary files like `env/`, `.tox`, `.coverage`, and `__pycache__`.

```
make clean
```

make test

Run unittests under the virtualenv's default Python version. Does not test all support Python versions. To test all supported versions, see make test-full.

```
make test
```

make test-full

Run unittest and linting for all supported Python versions. **NOTE:** This will fail if you do not have all Python versions installed on your system. If you are on an Ubuntu based system, the Dead Snakes PPA is a good resource for easily installing multiple Python versions. If for whatever reason you're unable to have all Python versions on your development machine, note that Travis-CI will run full integration tests on all pull requests.

```
make test-full
```

make lint

Run `make pylint` and `make pep8` commands.

```
make lint
```

make pylint

Run `pylint` compliance check on code base.

```
make pylint
```

make pep8

Run PEP8 compliance check on code base.

```
make pep8
```

make docs

Build documentation to `docs/_build/`.

```
make docs
```

CHAPTER 5

Indices and Tables

- *genindex*
- *modindex*
- *search*

C

E

F

G

I

S