Flask-And-Redis Documentation

Release 0.7

Igor Davydenko

Feb 14, 2019

Contents

1 Installation **3**

2 Usage **5**

3 Configuration **7**
- 3.1 REDIS_CLASS . . . 7
- 3.2 REDIS_URL . . . 8
- 3.3 Config prefix . . . 8

4 API **9**

5 Changelog **11**
- 5.1 1.0 (2019-02-14) . . . 11
- 5.2 0.7 (2016-11-12) . . . 11
- 5.3 0.6 (2015-01-08) . . . 11
- 5.4 0.5 (2013-05-10) . . . 11
- 5.5 0.4 (2012-09-29) . . . 12
- 5.6 0.3.3 (2012-08-29) . . . 12
- 5.7 0.3.2 (2012-08-15) . . . 12
- 5.8 0.3.1 (2012-06-19) . . . 12
- 5.9 0.3 (2012-05-21) . . . 12
- 5.10 0.2.1 (2012-03-30) . . . 12
- 5.11 0.2 (2012-03-30) . . . 12
- 5.12 0.1 (2012-03-12) . . . 12

6 Authors **13**

Python Module Index **15**

Flask-And-Redis provides simple as dead support of Redis database for Flask applications. Extension built around beautiful redis-py library by Andy McCurdy.

- Works on Python 2.7, 3.4+
- BSD licensed
- Latest documentation on Read the Docs
- Source, issues and pull requests on GitHub

Note: `Flask-And-Redis` named as is, cause `Flask-Redis` name already taken, but that library didn't match my needs.

CHAPTER 1

Installation

Use pip to install Flask-And-Redis to your system or virtual environment:

```
$ pip install Flask-And-Redis
```

Otherwise you could download source dist from GitHub or PyPI and put `flask_redis.py` file somewhere to `$PYTHONPATH`, but this way is not recommended. Use pip for all good things.

CHAPTER 2

Usage

In regular case all you need is importing *Redis* instance and initialize it with `app` instance, like:

```
from flask import Flask
from flask_redis import Redis

app = Flask(__name__)
redis = Redis(app)
```

But if you use application factories you could use *init_app()* method,

```
redis = Redis()
# The later on
app = create_app('config.cfg')
redis.init_app(app)
```

Also later you can get `redis` connection from `app.extensions['redis']` dict, where `key` is config prefix and `value` is configured redis connection.

CHAPTER 3

Configuration

`Flask-And-Redis` understands all keyword arguments which should be passed to `redis.StrictRedis` or `redis.Redis` classes init method. In easiest way all you need is putting

- `REDIS_HOST`
- `REDIS_PORT`
- `REDIS_DB`

to your settings module. Other available settings are:

- `REDIS_PASSWORD`
- `REDIS_SOCKET_TIMEOUT`
- `REDIS_CONNECTION_POOL`
- `REDIS_CHARSET`
- `REDIS_ERRORS`
- `REDIS_DECODE_RESPONSES`
- `REDIS_UNIX_SOCKET_PATH`

Later these values would initialize redis connection and all public methods of connection's instance would be copied to *`Redis`*. Also connection would be stored in `Redis.connection` attribute and `app.extensions['redis']` dict.

In addition extension has two more configuration options and ability to connect to multiple redis databases.

3.1 REDIS_CLASS

New in version 0.5.

Before 0.5 version only `redis.Redis` connection used. But as times change and `redis.StrictRedis` class grab default status we start to using it as our default connection class.

To change this behavior or even use your own class for redis connection you should pass a class itself or its path to `REDIS_CLASS` setting as:

```
from redis import Redis
REDIS_CLASS = Redis
```

or:

```
REDIS_CLASS = 'redis.Redis'
REDIS_CLASS = 'path.to.module.Redis'
```

3.2 REDIS_URL

New in version 0.2.

Sometimes, your redis settings stored as `redis://...` url (like in Heroku or DotCloud services), so you could to provide just `REDIS_URL` setting and `Flask-And-Redis` auto parsed that value and will configure then valid redis connection.

In case, when `REDIS_URL` provided all appropriate configurations, and other keys are overwritten using their values at the present URI.

3.3 Config prefix

New in version 0.4.

Config prefix allows you to determine the set of configuration variables used to configure `redis.Redis` connection. By default, config prefix `REDIS` would be used.

But when you want to initialize multiple redis connections, you could do this like:

```
from flask import flask
from flask.ext.redis import Redis

app = Flask(__app__)
app.config['REDIS_HOST'] = 'localhost'
app.config['REDIS_PORT'] = 6379
app.config['REDIS_DB'] = 0
redis1 = Redis(app)

app.config['REDIS2_URL'] = 'redis://localhost:6379/1'
redis2 = Redis(app, 'REDIS2')
```

CHAPTER 4

API

class `flask_redis.`**`Redis`**(*app=None*, *config_prefix=None*)

Simple as dead support of Redis database for Flask apps.

`__init__`(*app=None*, *config_prefix=None*)

Initialize Redis extension for Flask application.

If `app` argument provided then initialize redis connection using application config values.

If no `app` argument provided you should do initialization later with *`init_app()`* method.

Generally extension expects configuration to be prefixed with `REDIS` config prefix, to customize things pass different `config_prefix` here or on calling *`init_app()`* method. For example, if you have URL to Redis in `CACHE_URL` config key, you should pass `config_prefix='CACHE'` to extension.

Parameters

- **app** – `flask.Flask` application instance.
- **config_prefix** – Config prefix to use. By default: `REDIS`

`connection`

Return Redis connection for current app.

`get_app`()

Get current app from Flast stack to use.

This will allow to ensure which Redis connection to be used when accessing Redis connection public methods via plugin.

`init_app`(*app*, *config_prefix=None*)

Actual method to read redis settings from app configuration, initialize Redis connection and copy all public connection methods to current instance.

Parameters

- **app** – `flask.Flask` application instance.
- **config_prefix** – Config prefix to use. By default: `REDIS`

CHAPTER 5

Changelog

5.1 1.0 (2019-02-14)

- Drop Python 2.6 & 3.3 support, ensure Python 3.6+ support. Kudos to jezdez for pull request & implementation
- Use *inspect.getfullargspec()* if available. Kudos to vibiu for pull request & implementation

5.2 0.7 (2016-11-12)

- Improve multiple app support. Kudos to timothyqiu for pull request & implementation
- Simplify running tests for test application

5.3 0.6 (2015-01-08)

- Python 3 support.
- Move documentation to Read the Docs.
- Refactor example test project to Comments app which shows how to use two Redis databases simultaneously.

5.4 0.5 (2013-05-10)

- Use `redis.StrictRedis` as connection class by default.
- Understands unix socket path in `REDIS_HOST`.
- Updates to README.

5.5 0.4 (2012-09-29)

- Big refactor for *Redis* class. Do not inherit `redis.Redis` class, store active redis connection in `Redis.connection` attribute and `app.extensions['redis']` dict.
- Add support of `config_prefix` keyword argument for *Redis* or *init_app()* methods.
- Support multiple redis connections in test application.

5.6 0.3.3 (2012-08-29)

- Fix problem while parsing `REDIS_URL` value, strip unnecessary slashes from database path (like `redis://localhost:6379/12/`).

5.7 0.3.2 (2012-08-15)

- Added `redis` as install requirement in `setup.py`.

5.8 0.3.1 (2012-06-19)

- Move from `flask_redis` package to python module.
- Little improvements for storing `_flask_app` attribute to *Redis* instance.

5.9 0.3 (2012-05-21)

- Implement *init_app()* method.

5.10 0.2.1 (2012-03-30)

- Convert `REDIS_PORT` to an `int` instance.

5.11 0.2 (2012-03-30)

- Added support of `REDIS_URL` setting. By default, *Redis* will try to guess host, port, user, password and db settings from that value.

5.12 0.1 (2012-03-12)

- Initial release.

CHAPTER 6

Authors

- Igor Davydenko (original idea & code)
- Lexo Charles
- Hidetaka Yamashita
- Timothy Qiu
- vibiu
- Jannis Leidel

Python Module Index

f

flask_redis, 9

Index

Symbols

__init__() (flask_redis.Redis method), 9

C

connection (flask_redis.Redis attribute), 9

F

flask_redis (module), 9

G

get_app() (flask_redis.Redis method), 9

I

init_app() (flask_redis.Redis method), 9

R

Redis (class in flask_redis), 9