sprockets.clients.redis

Release 1.2.0

July 19, 2016

Contents

Base functionality for accessing/modifying data in Redis

CHAPTER 1

Installation

sprockets.clients.redis is available on the Python Package Index and can be installed via pip or easy_install:

```
pip install sprockets.clients.redis
```

CHAPTER 2

Requirements

- consistent_hash==1.0
- hiredis==0.1.6
- redis==2.10.3

CHAPTER 3

Simple Example

Simple CRUD operations on a key.

```
>>> import os
>>> os.environ['REDIS_URI'] = 'redis://localhost/'

>>> shard = ShardedRedisConnection()
<sprockets.clients.redis.ShardedRedisConnection at 0x1046f2c90>

>>> shard.set('foo', 1)
>>> shard.get('foo')
'1'
>>> shard.delete('foo')
>>> value = shard.get('foo')
>>> value is None
True
```

Setting a TTL on your key.

```
>>> import os
>>> import time
>>> os.environ['REDIS_URI'] = 'redis://localhost/'

>>> shard = ShardedRedisConnection()
<sprockets.clients.redis.ShardedRedisConnection at 0x1046f2c90>

>>> shard.set('bar', 1, ttl=2)
>>> shard.get('bar')
'1'
>>> time.sleep(2)
>>> value = shard.get('foo')
>>> value is None
True
```

CHAPTER 4

API Documentation

4.1 clients.redis

Base functionality for accessing/modifying data in Redis. Currently the supported functionality is accessing Redis in a sharded manner.

class sprockets.clients.redis.**RedisConnection**
Maintain a Redis connection.

This class establishes a redis connection based off of the IP address resolved from the `hostname` part of the `REDIS_URI` environment variable

Note: The hostname in the `REDIS_URI` will be DNS resolved and a connection will be opened for the address returned in the answer section.

class sprockets.clients.redis.**ShardedRedisConnection**
Maintain a list of several Redis backends in a sharded manner.

This class establishes several pools based off of the IP addresses resolved from the `hostname` part of the `REDIS_URI` environment variable. Any reads, writes, or deletes will be delegated to the proper Redis backend by determining which shard the query should be directed to.

Additionally, the `info` method is available to gather health information across all of the servers in the backend. This data can be used to determine the health of the service.

Note: The hostname in the `REDIS_URI` will be DNS resolved and a connection will be opened for each address returned in the answer section. You can specify a Round-Robin DNS record and we will open a connection to all hosts returned.

delete(*key*)
Delete a `key` in a Redis shard.

get(*key*)
Get a `key` in a Redis shard.

info()
Return a list of the health of each connected redis server.

Return type list

Returns A list of the server info from all of the server shards.

sadd(*key*, **values*)
Add the specified `values` to the set stored at `key`.

set(*key*, *value*, *ttl=None*)
Set `key` to `value` in a Redis shard.

sismember(*key*, *member*)
Returns if `member` is a member of the set stored at `key`.

smembers(*key*)
Return all members of the set stored at `key`.

4.2 Release History

4.2.1 Next Release

4.2.2 '2.0.0'_ (2015-06-22)

- Backwards incompatible change to the `RedisConnection` class. Inherit from `StrictRedis` instead of wrapping it.

4.2.3 '1.2.0'_ (2015-05-18)

- Add a RedisConnection that is not sharded.

4.2.4 1.1.0 (2015-05-18)

- Add support for some set based functionality the redis client provides

4.2.5 1.0.1 (2015-04-30)

- Don't log the payload in the DEBUG level logs, causes an encoding error when it's not ascii

4.2.6 1.0.0 (2015-03-30)

- Initial release of the sharded redis connection.

4.3 How to Contribute

Do you want to contribute fixes or improvements?

AWesome! *Thank you very much, and let's get started.*

4.3.1 Set up a development environment

The first thing that you need is a development environment so that you can run the test suite, update the documentation, and everything else that is involved in contributing. The easiest way to do that is to create a virtual environment for your endevours:

```
$ pyvenv env
```

Don't worry about writing code against previous versions of Python unless you you don't have a choice. That is why we run our tests through tox. If you don't have a choice, then install virtualenv to create the environment instead. The next step is to install the development tools that this project uses. These are listed in *dev-requirements.txt*:

```
$ env/bin/pip install -qr dev-requirements.txt
```

At this point, you will have everything that you need to develop at your disposal. *setup.py* is the swiss-army knife in your development tool chest. It provides the following commands:

./setup.py nosetests Run the test suite using nose and generate a nice coverage report.

./setup.py build_sphinx Generate the documentation using sphinx.

./setup.py flake8 Run flake8 over the code and report style violations.

If any of the preceding commands give you problems, then you will have to fix them **before** your pull request will be accepted.

4.3.2 Running Tests

The easiest (and quickest) way to run the test suite is to use the *nosetests* command. It will run the test suite against the currently installed python version and report not only the test result but the test coverage as well:

```
$ ./setup.py nosetests

running nosetests
running egg_info
writing dependency_links to sprockets.clients.redis.egg-info/dependency_links.txt
writing top-level names to sprockets.clients.redis.egg-info/top_level.txt
writing sprockets.clients.redis.egg-info/PKG-INFO
reading manifest file 'sprockets.clients.redis.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no previously-included files matching '__pycache__'...
warning: no previously-included files matching '*.swp' found ...
writing manifest file 'sprockets.clients.redis.egg-info/SOURCES.txt'
...

Name                        Stmts   Miss Branch BrMiss  Cover   Missing
-----------------------------------------------------------------------
...
-----------------------------------------------------------------------
TOTAL                          95      2     59      2    97%
-----------------------------------------------------------------------
Ran 44 tests in 0.054s

OK
```

That's the quick way to run tests. The slightly longer way is to run the detox utility. It will run the test suite against all of the supported python versions in parallel. This is essentially what Travis-CI will do when you issue a pull request anyway:

```
$ env/bin/detox
py27 recreate: /.../sprockets.clients.redis/build/tox/py27
GLOB sdist-make: /.../sprockets.clients.redis/setup.py
py33 recreate: /.../sprockets.clients.redis/build/tox/py33
py34 recreate: /.../sprockets.clients.redis/build/tox/py34
py27 installdeps: -rtest-requirements.txt, mock
py33 installdeps: -rtest-requirements.txt
py34 installdeps: -rtest-requirements.txt
py27 inst: /.../sprockets.clients.redis/build/tox/dist/sprockets.clients.redis-0.0.0.zip
py27 runtests: PYTHONHASHSEED='2156646470'
py27 runtests: commands[0] | /../sprockets.clients.redis/build/tox/py27/bin/nosetests
py33 inst: /../sprockets.clients.redis/.build/tox/dist/sprockets.clients.redis-0.0.0.zip
py34 inst: /../sprockets.clients.redis/.build/tox/dist/sprockets.clients.redis-0.0.0.zip
py33 runtests: PYTHONHASHSEED='2156646470'
py33 runtests: commands[0] | /.../sprockets.clients.redis/build/tox/py33/bin/nosetests
py34 runtests: PYTHONHASHSEED='2156646470'
py34 runtests: commands[0] | /.../sprockets.clients.redis/build/tox/py34/bin/nosetests
_____________________________________ summary _____________________________________
  py27: commands succeeded
  py33: commands succeeded
  py34: commands succeeded
  congratulations :)
```

This is what you want to see. Now you can make your modifications and keep the tests passing.

4.3.3 Submitting a Pull Request

Once you have made your modifications, gotten all of the tests to pass, and added any necessary documentation, it is time to contribute back for posterity. You've probably already cloned this repository and created a new branch. If you haven't, then checkout what you have as a branch and roll back *master* to where you found it. Then push your repository up to github and issue a pull request. Describe your changes in the request, if Travis isn't too annoyed someone will review it, and eventually merge it back.

CHAPTER 5

Version History

See Release History

CHAPTER 6

Contributing

Issues and Pull Requests are always welcome. For more information on how to contribute please refer to How to Contribute.

CHAPTER 7

Issues

Please report any issues to the Github project at https://github.com/sprockets/sprockets.clients.redis/issues

CHAPTER 8

Source

`sprockets.clients.redis` source is available on Github at https://github.com/sprockets/sprockets.clients.redis

CHAPTER 9

License

`sprockets.clients.redis` is released under the 3-Clause BSD license.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

S

D

G

I

R

S