wcraas$_{s}torageDocumentation$

Release 0.1.3

Kolokotronis Panagiotis

Oct 18, 2020

Contents:

CHAPTER 1

wcraas_storage

WCraaS Storage Service

- Free software: MIT license
- Documentation: https://wcraas-storage.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install wcraas_storage, run this command in your terminal:

```
$ pip install wcraas_storage
```

This is the preferred method to install wcraas_storage, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for wcraas_storage can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/WCraaS/wcraas_storage
```

Or download the tarball:

```
$ curl -OJL https://github.com/WCraaS/wcraas_storage/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use wcraas_storage in a project:

```
import wcraas_storage
```

CHAPTER 4

wcraas_storage

4.1 wcraas_storage package

4.1.1 Submodules

4.1.2 wcraas_storage.cli module

Console script for wcraas_storage.

4.1.3 wcraas_storage.config module

class wcraas_storage.config.**Config**

Bases: *wcraas_storage.config.Config*

classmethod fromenv()

Create a *wcraas_storage.Config* from Environment Variables.

```
>>> conf = Config.fromenv()
>>> type(conf)
<class 'config.Config'>
>>> conf._fields
('amqp', 'mongo', 'mapping', 'loglevel')
>>> conf.amqp
AMQPConfig(host='localhost', port=5672, user='guest', password='guest')
>>> conf.mongo
mongo = MongoConfig(host='localhost', port=27017, db='wcraas', user=None,␣
↪password=None)
>>> conf.mapping
{}
>>> conf.loglevel
20
```

class wcraas_storage.config.**MongoConfig**
Bases: *wcraas_storage.config.MongoConfig*

classmethod fromenv()
Create a *wcraas_storage.MongoConfig* from Environment Variables.

```
>>> conf = MongoConfig.fromenv()
>>> type(conf)
<class 'config.MongoConfig'>
>>> conf._fields
('host', 'port', 'db', 'user', 'password')
>>> conf.host
'localhost'
>>> conf.port
27017
```

4.1.4 wcraas_storage.wcraas_storage module

The WCraaS Storage module is responsible for providing storage services for the platform.

class wcraas_storage.wcraas_storage.**StorageWorker**(*amqp: wcraas_common.config.AMQPConfig, mongo: wcraas_storage.config.MongoConfig, mapping: Dict[str, str], loglevel: int, *args, **kwargs*)
Bases: wcraas_common.wcraas_common.WcraasWorker

get_queue_by_collection(*collection: str*) → str
Return the queue that corresponds to the given collection.

Parameters **collection** (*string*) – The collection with which to determine the queue.

list_collections() → Dict[str, List[Dict[str, str]]]
AMQP function that lists available collections in selected MongoDB.

run() → None
Helper function implementing the synchronous boilerplate for initilization and teardown.

start() → None
Asynchronous runtime for the worker, responsible of managing and maintaining async context open.

store(*message: aio_pika.message.IncomingMessage*) → None
AMQP consumer function, that inserts an *IncomingMessage*'s json-loaded body in a MongoDB collection based on the source exchange.

Parameters **message** (*aio_pika.IncomingMessage*) – The message that trigered the consume callback.

4.1.5 Module contents

Top-level package for storage.

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/WCraaS/wcraas_storage/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.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/WCraaS/wcraas_storage/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 :)

5.2 Get Started!

Ready to contribute? Here's how to set up *wcraas_storage* for local development.

1. Fork the *wcraas_storage* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/wcraas_storage.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 wcraas_storage
$ cd wcraas_storage/
$ 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 wcraas_storage tests
$ python setup.py test or pytest
$ 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.

5.3 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 3.6 and 3.7, and for PyPy. Check https://travis-ci.org/WCraaS/wcraas_storage/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_wcraas_storage
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Kolokotronis Panagiotis <panagiks@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.3 (2019-10-28)

- User common module's pre-implemented patterns.
- Ensure common module's base version.

7.2 0.1.1 (2019-10-27)

- Add *list_collections* RPC function.

7.3 0.1.0 (2019-09-21)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index