BlanketDB Documentation

Release 0.4.0

luphord

Oct 19, 2020

Contents:

1 BlanketDB **1**
- 1.1 Features . . . 1
- 1.2 Quickstart . . . 2
- 1.3 Credits . . . 2

2 Installation **3**
- 2.1 Stable release . . . 3
- 2.2 From sources . . . 3
- 2.3 Single File . . . 3

3 Usage **5**
- 3.1 Command line . . . 5
- 3.2 Python . . . 5

4 Web interface **7**
- 4.1 Create entries . . . 7
- 4.2 Retrieve entries . . . 8
- 4.3 Query database . . . 8
- 4.4 Delete entries . . . 10

5 blanketdb **13**
- 5.1 blanketdb module . . . 13
- 5.2 blanketdb_wsgi module . . . 14
- 5.3 setup module . . . 14
- 5.4 tests package . . . 14

6 Contributing **15**
- 6.1 Types of Contributions . . . 15
- 6.2 Get Started! . . . 16
- 6.3 Pull Request Guidelines . . . 17
- 6.4 Tips . . . 17
- 6.5 Deploying . . . 17

7 Credits **19**
- 7.1 Development Lead . . . 19
- 7.2 Contributors . . . 19

8 History **21**
8.1 0.4.0 (2020-02-26) . . . 21
8.2 0.3.4 (2020-02-26) . . . 21
8.3 0.3.3 (2019-12-12) . . . 21
8.4 0.3.2 (2019-12-04) . . . 21
8.5 0.3.1 (2019-03-06) . . . 21
8.6 0.3 (2019-03-06) . . . 22
8.7 0.2.2 (2019-01-31) . . . 22
8.8 0.2.1 (2019-01-24) . . . 22
8.9 0.2.0 (2019-01-24) . . . 22
8.10 0.1.0 (2019-01-18) . . . 22

9 Indices and tables **23**

Python Module Index **25**

Index **27**

CHAPTER 1

BlanketDB

BlanketDB is a very simple database written in Python based on SQLite. It is intended for small IoT projects where you need a quick way to collect and store data from sensors and other devices. You communicate to BlanketDB using HTTP GET / POST / DELETE requests. Request and response bodies are usually JSON, but you can also POST HTML forms directly to BlanketDB. There is no schema in the database, you simply store objects in buckets.

BlanketDB is free software provided under a MIT license. Documentation is available at https://blanketdb.readthedocs.io.

Why is it called BlanketDB? Well, a blanket is simple, lightweight, portable and keeps you warm. But if you really want to relax, you'll need a couch.

1.1 Features

- GET / POST / DELETE requests to communicate with BlanketDB
- JSON requests / responses
- HTML forms can POST directly to BlanketDB
- Data stored in buckets
- Schemaless
- Query using various parameters to a HTTP GET request
- Data is stored in a single file on the file system which is a SQLite database
- BlanketDB is a single Python file without any dependencies besides the standard library

- No sequrity whatsoever; BlanketDB is completely open to readers and writers (use with care!)

1.2 Quickstart

To install BlanketDB, you'll need a Python (>=3.4) installation with pip:

```
$ pip install blanketdb
```

To use BlanketDB as a standalone database (and communicate over HTTP), enter the following command:

```
$ python3 -m blanketdb -i localhost -p 8080 -f /path/to/db.sqlite
```

BlanketDB will now serve its web interface at http://localhost:8080. You can open this page in your browser to check if everything works.

To use BlanketDB in a Python project, enter the following code:

```
from blanketdb import BlanketDB
db = BlanketDB('/path/to/db.sqlite')

# you can now use db using its Python API
db.store_dict(x='test')['id']
for entry in db:
    print(entry)

# db is alse a wsgi conforming callable
# you can use it e.g. with the wsgi reference implementation
from wsgiref.simple_server import make_server
httpd = make_server('localhost', 8080, db)
httpd.serve_forever()
```

Detailed documentation is available at https://blanketdb.readthedocs.io.

1.3 Credits

Main author and project maintainer is luphord.

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

CHAPTER 2

Installation

2.1 Stable release

To install BlanketDB, run this command in your terminal:

```
$ pip install blanketdb
```

This is the preferred method to install BlanketDB, 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 BlanketDB can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/luphord/blanketdb
```

Or download the tarball:

```
$ curl  -OL https://github.com/luphord/blanketdb/tarball/master
```

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

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

2.3 Single File

BlanketDB has no dependencies besides the Python standard library. You can also just grab blanketdb.py and drop it into your project.

CHAPTER 3

Usage

3.1 Command line

To use BlanketDB as a standalone database (and communicate over HTTP), enter the following command:

```
$ python3 -m blanketdb -i localhost -p 8080 -f /path/to/db.sqlite
```

BlanketDB will now serve its web interface at http://localhost:8080. You can open this page in your browser to check if everything works.

The following command line options are available:

```
usage: blanketdb.py [-h] [-i INTERFACE] [-p PORT] [-f FILE]

Start a BlanketDB instance using wsgiref.simple_server.

optional arguments:
-h, --help            show this help message and exit
-i INTERFACE, --interface INTERFACE
                        Interface to listen on
-p PORT, --port PORT  Port to listen on
-f FILE, --file FILE  Database file to use
```

3.2 Python

To use BlanketDB in a project, enter the following Python code:

```
from blanketdb import BlanketDB
db = BlanketDB('/path/to/db.sqlite')

# you can now use db using its Python API
db.store_dict(x='test')['id']
```

(continues on next page)

(continued from previous page)

```
for entry in db:
    print(entry)

# db is alse a wsgi conforming callable
# you can use it e.g. with the wsgi reference implementation
from wsgiref.simple_server import make_server
httpd = make_server('localhost', 8080, db)
httpd.serve_forever()
```

You may want to check the Python API of BlanketDB.

CHAPTER 4

Web interface

This section documents the Web interface (HTTP API) of BlanketDB. For simplicity we assume that BlanketDB is served at http://localhost:8080.

4.1 Create entries

To create an entry in the default bucket perform this request:

```
POST http://localhost:8080/
```

The body of the POST request may either be an arbitrary JSON object, e.g.:

```
{
    "a": 1.23,
    "test": "somedata"
}
```

or URL-encoded form content (as is created by standard HTML form submission), e.g.:

```
a=1.23&test=somedata
```

In both cases, BlanketDB will answer with a JSON object like:

```
{
    "id": 3,
    "bucket": "default",
    "timestamp": "2019-01-23T17:11:41.168836",
    "data": {
        "a": 1.23,
        "test": "somedata"
    }
}
```

If you want to store to a bucket named *mybucket*, post to this URL:

```
POST http://localhost:8080/mybucket
```

4.2 Retrieve entries

To retrieve an individual entry using its ID (e.g. 123), use the following request:

```
GET http://localhost:8080/_entry/123
```

BlanketDB will answer with a response similar to the post request above:

```
{
    "id": 123,
    "bucket": "default",
    "timestamp": "2019-01-24T06:31:36.328127",
    "data": {
        "a": 1.23,
        "test": "somedata"
    }
}
```

If no entry with the given ID exists, BlanketDB will respond with a *404 Not found* HTTP error code. Note that you do not specify the bucket of the entry in the URL. IDs are unique across all buckets in BlanketDB.

The response above shows the data stored in entry 123 as well as corresponding metadata such as *timestamp* (of creation) and *bucket*. In order to make BlanketDB omit any metadata use

```
GET http://localhost:8080/_entry/123?meta=false
```

which will result in this reponse:

```
{
    "a": 1.23,
    "test": "somedata"
}
```

4.3 Query database

BlanketDB allows you to query entries using these filters:

- *bucket*
- *since_id* entries with the given ID (inclusive) or higher
- *before_id* entries with an ID lower than the given one (exclusive)
- *since* entries created at the given time (inclusive) or later
- *before* entries created before the given time (exclusive)

The *bucket* is specified in the URL, the remaining filters are given as query parameters. *since* and *before* can be specified as timestamps (e.g. “2019-01-24T06:52:06.181786” or just “2019-01-24”) or as multiples of seconds, minutes or hours (e.g. “2sec”, “7s”, “3min”, “8m”, “1hour”, “2hours”, “3h”).

In order to query all entries of bucket *mybucket* of the last two hours, use this request:

```
GET http://localhost:8080/mybucket?since=2hours
```

BlanketDB will respond in this form:


```
{
    "bucket_requested": "mybucket",
    "since_id": 0,
    "since": "2019-01-24T04:59:37.925981",
    "before_id": null,
    "before": null,
    "number_of_entries": 2,
    "last_id": 4,
    "limit": null,
    "newest_first": true,
    "entries": [
        {
            "id": 4,
            "bucket": "mybucket",
            "timestamp": "2019-01-24T06:59:30.462450",
            "data": {
                "b": 1.23,
                "test": "somedata2"
            }
        },
        {
            "id": 3,
            "bucket": "mybucket",
            "timestamp": "2019-01-24T06:59:23.005946",
            "data": {
                "a": 1.23,
                "test": "somedata"
            }
        }
    ]
}
```


In the same way as retrieving individual entries you can omit entry metadata using

```
GET http://localhost:8080/mybucket?since=2hours&meta=false
```

which will result in:


```
{
    "bucket_requested": "mybucket",
    "since_id": 0,
    "since": "2019-01-24T05:00:02.552377",
    "before_id": null,
    "before": null,
    "number_of_entries": 2,
    "last_id": 4,
    "limit": null,
    "newest_first": true,
    "entries": [
        {
            "b": 1.23,
            "test": "somedata2"
        },
```


(continues on next page)

(continued from previous page)

```
        {
            "a": 1.23,
            "test": "somedata"
        }
    ]
}
```

If you want to limit the number of entries retrieved, you can specify the *limit* parameter. In this context you will likely want to specify whether you are interested in the oldest or newest entries. To query the latest 3 entries in *mybucket*, use the following request (without metadata for brevity)

```
GET http://localhost:8080/mybucket?meta=false&limit=3&newest_first=true
```

which will result in something like this:

```
{
    "bucket_requested": "mybucket",
    "since_id": 0,
    "since": null,
    "before_id": null,
    "before": null,
    "number_of_entries": 3,
    "last_id": 6,
    "limit": 3,
    "newest_first": true,
    "entries": [
        {
            "b": 1.23,
            "test": "somedata2"
        },
        {
            "b": 1.23,
            "test": "somedata2"
        },
        {
            "b": 1.23,
            "test": "somedata2"
        }
    ]
}
```

If *newest_first* is not specified, it will default to *true* (hence the example above would work without *newest_first*).

In order to paginate entries you can use a combination of *since_id* and *limit*. For each subsequent request you would read the *last_id* field of the response, icrement by 1 and then use that number as the new *since_id*.

4.4 Delete entries

You can delete individual entries using the following request (for entry 123):

```
DELETE http://localhost:8080/_entry/123
```

In addition, you can apply the query filters above when deleting entries. For example, to delete all entries before today you would use the request:

```
DELETE http://localhost:8080/?before=today
```

BlanketDB will respond with the usual query metadata and a field containing the number of entries deleted:


```
{
    "bucket_requested": null,
    "since_id": 0,
    "since": null,
    "before_id": null,
    "before": "2019-01-24",
    "number_of_entries_deleted": 3
}
```

CHAPTER 5

blanketdb

5.1 blanketdb module

A simple HTTP accessible database for IoT projects.

class blanketdb.**BlanketDB**(*connection_string: str*, *now: Callable[[], datetime.datetime] = <built-in method now of type object>*)

Bases: object

A simple HTTP accessible database for IoT projects

delete(*bucket: str = None*, *since_id: Optional[int] = None*, *since: Union[str, datetime.datetime, datetime.date, None] = None*, *before_id: Optional[int] = None*, *before: Union[str, datetime.datetime, datetime.date, None] = None*) → Any

Delete entries from this *BlanketDB* instance using various filters. *since* and *since_id* are inclusive, *before* and *before* are exclusive regarding the specified value.

query(*bucket: str = None*, *since_id: Optional[int] = None*, *since: Union[str, datetime.datetime, datetime.date, None] = None*, *before_id: Optional[int] = None*, *before: Union[str, datetime.datetime, datetime.date, None] = None*, *limit: int = -1*, *newest_first: bool = True*) → Iterable[Dict[str, Any]]

Query this *BlanketDB* instance using various optional filters. *since* and *since_id* are inclusive, *before* and *before* are exclusive regarding the specified value.

store(*data: Any*, *bucket: str = 'default'*) → Dict[str, Any]

Serialize *data* to json and store it under *bucket*.

store_dict(*bucket: str = 'default'*, ***kwargs*) → Dict[str, Any]

Serialize key word args to json and store under *bucket*.

blanketdb.**cli**() → None

5.2 blanketdb_wsgi module

5.3 setup module

5.4 tests package

5.4.1 Submodules

5.4.2 tests.test_blanketdb module

5.4.3 Module contents

Unit test package for blanketdb.

CHAPTER 6

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:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/luphord/blanketdb/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.

6.1.2 Fix Bugs

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

6.1.3 Implement Features

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

6.1.4 Write Documentation

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

6.1.5 Submit Feedback

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

6.2 Get Started!

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

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

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

6.3 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 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/luphord/blanketdb/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_blanketdb
```

6.5 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:

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

Travis will then deploy to PyPI if tests pass.

CHAPTER 7

Credits

7.1 Development Lead

- luphord <luphord@protonmail.com>

7.2 Contributors

None yet. Why not be the first?

CHAPTER 8

History

8.1 0.4.0 (2020-02-26)

- Start uwsgi using http protocol by default in DOCKERFILE (s.t. standalone use is possible)
- Overwrite CMD in docker-compose file to communicate via uwsgi protocol between nginx and blanketdb container

8.2 0.3.4 (2020-02-26)

- Support Python 3.8

8.3 0.3.3 (2019-12-12)

- Split tests into Python and HTTP API tests
- Added tests that can be executed against an actual HTTP API of *BlanketDB*

8.4 0.3.2 (2019-12-04)

- Release to trigger build on dockerhub

8.5 0.3.1 (2019-03-06)

- Improved clarity with default values

8.6 0.3 (2019-03-06)

- Type annotations for *BlanketDB*
- Python 3.4 is not supported anymore (as it does not know type annotations)

8.7 0.2.2 (2019-01-31)

- setuptools entrypoint for cli
- quickstart documentation
- added logo

8.8 0.2.1 (2019-01-24)

- fix tag confusion

8.9 0.2.0 (2019-01-24)

- Added CLI for starting *BlanketDB* with *wsgiref.simple_server*
- Tests for *BlanketDB* Web API using *webtest*
- Added documentation for usage and Web API

8.10 0.1.0 (2019-01-18)

- First release on PyPI.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

blanketdb, 13

t

tests, 14

Index

B

BlanketDB (*class in blanketdb*), 13
blanketdb (*module*), 13

C

cli() (*in module blanketdb*), 13

D

delete() (*blanketdb.BlanketDB method*), 13

Q

query() (*blanketdb.BlanketDB method*), 13

S

store() (*blanketdb.BlanketDB method*), 13
store_dict() (*blanketdb.BlanketDB method*), 13

T

tests (*module*), 14