Brush Documentation

Release 1.2.0

Michael V. DePalatis

July 18, 2016

Contents

A tool for logging and monitoring data from Menlo Systems optical frequency combs.

CHAPTER 1

Installation

Brush is on PyPI:

```
$ pip install brush
```

Python versions of at least 3.3 are recommended, though Brush should work with Python 2.7 if necessary.

Data is stored using a SQL database. PostgresQL is recommended, but any database supported by SQLAlchemy will work.

A database driver library will also need to be installed unless SQLite is used. For example, if using Postgres, the psycopg2 driver should be installed. On Debian-based Linux systems the following commands can be used:

```
$ sudo apt-get install -y postgresql-server-dev-all
$ pip install psycopg2
```

See the SQLAlchemy dialect documentation for additional details.

CHAPTER 2

Usage

Brush defines the following command-line options for collecting data:

```
  --config                        Path to configuration file (default
                                  ~/.brush.conf)
  --debug                         Enable debug output (default False)
  --offline                       Run in offline mode (default False)
  --redis-host                    Redis hostname (default localhost)
  --redis-password                Redis password
  --redis-port                    Redis port (default 6379)
  --save-when-unlocked            Write data to database when comb is unlocked
                                  (default False)
  --server-port                   Port to serve on (default 8090)
  --server-url-prefix             URL prefix
  --sql-table                     SQL table name (default brush)
  --sql-url                       SQL database URL (default
                                  sqlite:///brush.sqlite)
  --xmlrpc-host                   XMLRPC server hostname
  --xmlrpc-password               XMLRPC server password
  --xmlrpc-port                   XMLRPC server port (default 8123)
  --xmlrpc-user                   XMLRPC server user
```

These can also be written into a configuration file. For example:

```
xmlrpc_host = "localhost"
xmlrpc_port = 8123
xmlrpc_user = None
xmlrpc_password = None

sql_url = "sqlite://"
```

These values are just normal Python variables and will be overridden by any command-line options passed. See the Tornado documentation for additional details on configuration files.

With these settings stored in `brush.config`, Brush can then be started with the following command:

```
$ brush --config=brush.config
```

If using the default port, point your browser to `http://localhost:8090` and see the current comb status.

CHAPTER 3

HTTP API

Brush exposes the following routes for accessing data from the web user interface and other programs:

GET /
Render the web UI.

GET /data
Get data starting from the timestamp `start` up until the timestamp `stop`. Timestamps must be given as seconds since the epoch (i.e., Unix time) and passed as query arguments in the `GET` request.

If only `start` is given, the stop point is the current time.

Example:

```
http://localhost:8090/data?start=1465727734.4149404
```

Note: The database stores timestamps in UTC.

GET /data/current
Return the most recent data.

GET /data/recent
Return all data currently in the store.

GET /data/query/(.*)
Return the most recent value for the requested key.

GET /data/metadata
Return comb metadata.

Metadata includes types and descriptions of all data types.

GET /data/keys
Return all data keys.

GET /query/(.*)
Return the most recent value for the requested key.

CHAPTER 4

Contents

4.1 Comb API Reference

In addition to logging data from a comb and exposing an HTTP API for getting data, Brush also provides a Python API for communicating with the Menlo XMLRPC server.

class brush.comb.**DummyFrequencyComb**
Simulated frequency comb for testing purposes.

get_data()
Return random data in order to simulate the presence of a Menlo XMLRPC server. Only a small subset of possible data keys are provided.

Returns **data** : dict

randomized data values

get_data_since(*delta*)
Return random data from now - delta seconds ago.

class brush.comb.**FrequencyComb**(*host*, *port=8123*, *user=None*, *password=None*)
Class for communicating with a Menlo frequency comb.

Parameters **host** : str

server hostname

port : int

server port

user : str or None

username for authentication

password : str or None

password for authentication

get_data(*keys=[]*)
Query the XMLRPC server for the most recent data.

Parameters **keys** : list

List of keys to get. If empty, fetch all data.

Returns **result** : dict

All collected data

get_data_since(*delta*, *keys=[]*)
Get data since `delta` seconds ago.

get_version()
Return the version of the Menlo XMLRPC server software.

keys()
Return available data keys.

4.2 Development

This section contains notes of interest for those wishing to contribute to Brush.

4.2.1 Preliminaries

Additional tools for testing and documentation building can be installed with:

```
$ pip install -r requirements.txt
```

4.2.2 Documentation

Documentation is built using Sphinx. Either `cd` to the `docs` directory and run `make html` (`make.bat html` on Windows) or run `make docs` from the repository root directory.

4.2.3 Testing

Tests can be automated using pytest as the test runner:

```
$ py.test
```

in the repository root directory. Alternatively, tox may be used to further automate testing on multiple versions of Python:

```
$ tox
```

Some tests may require access to a real comb. The default behavior is to skip these tests so that tests can still be run on machines without comb access. In order to force them to run, run `py.test` with at least the environment variable `MENLO_COMB_HOST` defined to give the hostname for the computer controlling the frequency comb. Other valid environment variables for comb testing are `MENLO_COMB_PORT`, `MENLO_COMB_USER`, and `MENLO_COMB_PASSWORD`. When these are not given, the default values in the *brush.comb.FrequencyComb* constructor are used.

4.3 Changes

4.3.1 Version 1.2

2016-07-18

- Javascript is now bundled and minified using webassets, meaning that Node and NPM are no longer required. Note that the use of some ES2015 code means that the web interface may not work in older browsers.

- Reduce CPU usage with a real comb by limiting XMLRPC polling frequency.
- Ensure timestamps are always returned when getting data via the HTTP interface.
- Improve data downloading interface by using a datetime picker.

4.3.2 Version 1.1

2016-06-29

- Rework web user interface to use Vue.
- Implement charts with Chart.js.
- Use time zone aware timestamps on databases that support them. This breaks backwards compatibility: *timestamp* columns in existing databases must be manually altered to use (e.g., for Postgres, `TIMESTAMP WITH TIME ZONE` instead of `TIMESTAMP WITHOUT TIME ZONE`).

Updating existing tables with Postgres

To update the `timestamp` column to include time zones if using Postgres, you can issue the following SQL command:

```
ALTER TABLE brush
    ALTER COLUMN timestamp TYPE timestamp with time zone
    USING timestamp AT TIME ZONE 'UTC';
```

This update should be performed after stopping Brush. Good practice dictates that a backup should also be made prior to altering the table.

4.3.3 Version 1.0

2016-04-11

- Rework database: use all data from the XMLRPC server and optionally save data even when the comb is not locked.
- Simplify command-line usage: only data logging and basic web server for monitoring remain. Other features are still accessible via the `brush.comb` module to communicate with a frequency comb in other ways.

4.3.4 Version 0.2

(Unreleased, superseded by version 1.0.0)

- Added a web interface to view data and monitor status
- Improved command-line usage
- Storage to SQL databases

/

```
GET /,7
```

/data

```
GET /data,7
GET /data/current,7
GET /data/keys,7
GET /data/metadata,7
GET /data/query/(.*),7
GET /data/recent,7
```

/query

```
GET /query/(.*),7
```

D

F

G

K