Sick Muse Documentation

Release 0.2.0

Mark Lavin

January 17, 2013

CONTENTS

Sick Muse is an open source web application graphing RRD data stored by collectd.

CHAPTER

ONE

INSTALLATION DEPENDENCIES

Requires Python 2.6 or 2.7 and the following Python libraries:

- tornado >= 2.4 (Available under Apache v2)
- python-rrdtool >= 1.4 (Available under LGPL v3)

These dependencies do not ship with the library but will be resolved during the install:

```
pip install sickmuse
```

CHAPTER

TWO

RUNNING THE SERVER

Sick Muse runs on the Tornado webserver which is a single-threaded non-blocking server. Once installed you run this server using the `sickmuse` command:

```
sickmuse
```

This will start the server running on the default port `8282`. You can change the port using the `--port` option:

```
sickmuse --port=8080
```

CHAPTER

THREE

SCREENSHOTS

Figure 3.1: Homepage/Host Listing

Figure 3.2: Host Details

CHAPTER

FOUR

DOCUMENTATION

You can find a complete set of documentation on Read The Docs.

CHAPTER

FIVE

LICENSE

sickmuse is released under the BSD License. See the LICENSE file for more details.

In addition to the previously listed Python dependencies, this library makes use of the following projects which are included in the distribution:

- Twitter Bootstrap (Licensed under Apache v2)
- RequireJS (Licensed under BSD/MIT)
- jQuery (Licensed under MIT)
- Font Awesome (Licensed under CC BY 3.0)
- Flot (Licensed under MIT)
- Backbone (Licensed under MIT)
- Underscore (Licensed under MIT)

CHAPTER

SIX

CONTRIBUTING

This project is still in its early stages and there may be bugs or rapid changes to the internal APIs. If you think you've found a bug or are interested in contributing to this project check out sickmuse on Github.

Contents:

6.1 Installation Guide

This section will take you through installing and running the Sick Muse server. Before installing you should make sure you have Python 2.6 or 2.7 installed as well as setuptools or distribute. It is recommended to use pip and virtualenv as well.

The requirement of python-rrdtool needs to be compiled. Your system needs to have the appropriate header files prior to installation. For Ubuntu based systems you can install the required header files with `apt-get`:

```
sudo apt-get install libcairo2-dev libpango1.0-dev libglib2.0-dev libxml2-dev librrd-dev
```

For other operating systems you should use available package managers to install the headers for Python, libcairo2, libpango, libxml2, libglib2 and librrd.

6.1.1 Install

The recommended method for installing Sick Muse is using `pip` but you can also use `easy_install`:

```
pip install sickmuse
# or
easy_install sickmuse
```

6.1.2 Running the Server

Sick Muse runs on the Tornado webserver which is a single-threaded non-blocking server. Once installed you run this server using the `sickmuse` command:

```
sickmuse
```

This will start the server running on the default port `8282`. You can change the port using the `--port` option:

```
sickmuse --port=8080
```

See a *full list of configuration options*.

The server does not daemonize itself so it is recommneded that you use one of the many available daemonization tools such as Supervisord, Upstart, or Runit. An example Supervisor configuration is given below.

6.1.3 Example Supervisor Configuration

An example Supervisor configuration is given below. This file would be placed in `/etc/conf/supervisor/conf.d/sickmuse.conf`.

```
[program:sickmuse]
process_name=sickmuse
command=<path to install>/bin/sickmuse
numprocs=1
autostart=true
autorestart=true
```

`<path to install>` would be replaced by eiher the system bin if installed globally or the virtualenv bin directory if installed in a virtual environment. You can find this path via:

```
which sickmuse
```

6.1.4 Running Behind a Proxy

You may want to run Sick Muse behind a proxy such as Nginx or Apache to serve on its own domain or to enforce authentication. An example Nginx configuration is given to run this proxy. This file would be placed in both `/etc/nginx/sites-available/sickmuse.conf` and `/etc/nginx/sites-enabled/sickmuse.conf`.

```
upstream sickmuse {
    server 127.0.0.1:8282;
}

server {
    listen 80;
    server_name sickmuse.example.com;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_buffering on;
        proxy_intercept_errors on;
        proxy_pass http://sickmuse;
    }
}
```

6.2 Project Overview

The Sick Muse project was started to provide a simple alternative to the graphs generated by RRDTool. While there are a number of great alternative dashboards for the Graphite project, there aren't very many which work directly on top of Collectd's round-robin database files.

6.2.1 Project Goals

The primary goal of this project is ease of use. Sick Muse should be easy to install and run, easy to navigate, and easy to customize. Sick Muse should be performant enough for large projects but simple enough for small projects. While not all of these goals have been met they are the primary drivers for features planned in the future.

6.2.2 Project Components

On the server side, Sick Muse makes use of the Tornado web server because it provides great performance with a small resource footprint. It provides a complete solution for the needs of this project (url routing, template rendering, static resources) including the possibily of future real-time data via websockets.

On the client side, this project uses Backbone to query the API and maintain state on the client. The graphs are generated using the Flot library which has a well designed base API as well as a strong set of plugins for additional functionality.

The styles are based on Twitter Bootstrap. This gives sane default styles, including a responsive layout, along with easy customization using LESS.

6.3 Configuration Options

Below is a full set of configuration options for the Sick Muse server. These options are configured on the command line via `sickmuse --<name>=<value>`.

6.3.1 debug

Default: `False`

Turns the server into debug mode. In this mode the static resouces will use the non-compressed version and the Python code changes will auto reload. This setting is used for local development and should not be used in production.

6.3.2 port

Default: `8282`

The port on which the server runs.

6.3.3 prefix

Default: `""` (Empty String)

The `prefix` argument can be used to run the server on under a url prefix. For instance if you wanted to run the server under `sickmuse` you could start the server with:

```
sickmuse --prefix=sickmuse
```

and then proxy the server with Nginx

```
upstream sickmuse {
    server 127.0.0.1:8282;
}
```

```
server {
    listen 80;
    server_name example.com;

    location /sickmuse/ {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_buffering on;
        proxy_intercept_errors on;
        proxy_pass http://sickmuse;
    }
}
```

You could then see the server running at http://example.com/sickmuse/.

6.3.4 rrd_directory

Default: `/var/lib/collectd/rrd/`

This is the directory where the server parses the round robin database files. It expects this directory to have the structure created by Collectd where the sub-directories are organized `/<host-name>/<plugin-name>/<instance>.rrd`.

6.4 Contributing Guide

This guide documents ways to contribute to the Sick Muse project as well as the tools used to make that easier. The project source code and issue tracking is hosted on Github.

6.4.1 Getting the Source

You can clone the repository from Github:

```
git clone git://github.com/mlavin/sickmuse.git
```

However this checkout will be read only. If you want to contribute code you should create a fork and clone your fork. You can then add the main repository as a remote:

```
git clone git@github.com:<your-username>/sickmuse.git
git remote add upstream git://github.com//mlavin/sickmuse.git
git fetch upstream
```

6.4.2 Running the Debug Server

When running the server from inside the repo you can run the server via:

```
python -m sickmuse.app
```

Similarly you can change the port with the `--port` option:

```
python -m sickmuse.app --port=8000
```

Another helpful option for running locally is to use the `--debug` option:

```
python -m sickmuse.app --debug
```

This will auto-reload the Python modules as they are changed as well as make it easier to manage CSS and JS changes which is described below.

6.4.3 Installing JS/CSS Libraries

The production version of Sick Muse ships with minified versions of the CSS and JS built from and including its static dependencies. To develop locally you will need the full versions of these libraries.

You may use the built-in Makefile to grab these. Behind the scenes this will use bower package manager:

```
# Install bower via NPM
npm install bower
# Install libraries with bower
make install-static
```

6.4.4 Building the CSS

The CSS used by Sick Muse is built using LESS. No changes should be made to the `sickmuse.css` directly. Instead changes should be made to the `extensions.less` file. After changes are made to `extensions.less` you can create the new compressed CSS with the Node based complier:

```
# Install less from the NPM package
npm install less
# Build compressed CSS
make build-css
```

When the server is running in debug mode it uses the client-side LESS compiler to make local development easier.

6.4.5 Building the JS

The JS used by Sick Muse uses RequireJS to manage its dependency loading and building the bundled version. Similar to the CSS, no changes should be made directly to the `sickmuse-built.js`. To build the bundled JS you will need the `requirejs` optimizer:

```
# Install requirejs from the NPM package
npm install requirejs
# Build compressed JS
make build-js
```

When the server is running in debug mode it uses the non-compressed version for easier debugging.

This project using JSHint for JS style and detecting problematic JS patterns:

```
# Install jshint from the NPM package
npm install jshint
# Build compressed JS
make lint-js
```

6.4.6 Building the Documentation

The docs are written in ReST and built using Sphinx. As noted above you can use tox to build the documentation or you can build them on their own via:

```
tox -e docs
```

or:

```
make html
```

from inside the `docs/` directory.

6.4.7 Submitting a Pull Request

The easiest way to contribute code or documentation changes is through a pull request. For information on submitting a pull request you can read the Github help page https://help.github.com/articles/using-pull-requests.

Pull requests are a place for the code to be reviewed before it is merged. This review will go over the coding style as well as if it solves the problem indended and fits in the scope of the project. It may be a long discussion or it might just be a simple thank you.

Not necessarily every request will be merged but you should not take it personally if you change is not accepted. If you want to increase the chances of your change being incorporated then here are some tips.

- Address a known issue. Preference is given to a request that fixes a currently open issue.
- Include documentation and tests when appropriate. New features should be tested and documented. Bugfixes should include tests which demostrate the problem.
- Keep it simple. It's difficult to review a large block of code so try to keep the scope of the change small.

You should also feel free to ask for help writing tests or writing documentation if you aren't sure how to go about it.

6.5 Release History

Below is a list of all the public releases and a summary of the related changes in those releases.

6.5.1 v0.2.0 (Released 01-17-2013)

- Added project documentation (overview, sceenshots, installation, contributing guide)
- Started on server-side test suite and Travis CI integration
- Added `rrd_directory` configuration option
- Font Awesome upgraded to 3.0.0
- Added `prefix` configuration option
- Added graceful server shutdown
- Fixed issue with relative imports breaking server debug mode
- Fixed missing static resources when running in debug mode from PyPi install

6.5.2 v0.1.0 (Released 12-16-2012)

- Initial public release