hamster-dbus Documentation

Release 0.10.0

Eric Goller

Jun 02, 2017

Contents

Contents:

CHAPTER 1

hamster-dbus

A dbus interface to `hamster-lib`.

- Free software: GPL3
- Documentation: https://hamster-dbus.readthedocs.org.

1.1 How to use

`hamster-dbus` provides two very different but related functionalities.

1. `hamster_dbus.objects` provides several dbus object subclasses that can be used to export services over dbus that in effect expose `hamster-lib` to over dbus.
2. `hamster_dbus.storage` contains `DBusStore` which can be used as a valid backend for `hamster-lib` that can communicate with the objects defined in `hamster_dbus.objects`. This means any client that supports `hamster-lib` can use this backend (instead of the default SQLAlchemy one for example) to easily make their clients use an available dbus service instead of handling the backend functionality itself via SQLAlchemy.

These two aspects are independent of each other but are two opposing sides (server and client of sorts) of the same medal.

On top of this, a primitive example dbus-service executeable (`hamster_dbus_service.py`) has been included that can be used to get a minimal hamster-dbus service running in no time.

1.2 Testing & Coverage

The `hamster-dbus` project strives to provide maintainable, well documented and tested code. To this end we do provide a basic test suite that is actively maintained and aims to provide >90% coverage. Unfortunately we currently lack the insight into glib/dbus best practices with regards to testing and our current `pytest` based solution does only somewhat work. The main problem is providing an isolated environment for actual unit testing (not integration tests). The way we handle things right now is by providing a dedicated fixture that launches a separate session bus in a new

process that our "objects to be tested" get hooked into. While this works most of the time there are two practical issues here (besides not being proper unit tests):

1. You may see an error like this when running the test suite:

```
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has not \
    been called
[xcb] Aborting, sorry about that.

Whilst we do not really understand whats going on this is most likely due
to the fact that the new spawned session bus process is seperate from the
actual main look.
```

2. `coverage` will report most of the "object" code as untested despite various tests executing their methods. This may be because those methods are "shadowed" by the `@method` decorator. Again, we lack the insight to deal with this right now.

So if you have any hints, pointers or even PRs that can help us improving our test setup we would be most grateful! Until then we will not be able to automatically run the test suite on a CI server which greatly limits our QA :(

To run the test suite locally, just execute the following within your virtualenv (after `make develop`):

```
make test
```

1.2.1 Sidenote About Testing Signals

So far we have not managed to establish a proper way of testing signals. In order to manually check if they are emitted as expected you may use the following (`dbus-monitor` needs to be installed):

```
dbus-monitor "type='signal',sender='org.projecthamster.HamsterDBus',interface='org.
↪projecthamster.HamsterDBus1'
```

1.3 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-pypackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install hamster-dbus
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv hamster-dbus
$ pip install hamster-dbus
```

CHAPTER 3

Usage

To use hamster-dbus in a project:

```
import hamster-dbus
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://projecthamster.atlassian.net/projects/DBUS/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.

4.1.2 Fix Bugs

Look through the issues for bugs. Anything with the type "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the issues for features. Anything typed as a "story" or "task" is open to whoever wants to implement it.

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://projecthamster.atlassian.net/projects/DBUS/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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *hamster-dbus* for local development.

1. Fork the *hamster-dbus* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/hamster-dbus.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 hamster-dbus
$ cd hamster-dbus/
$ make 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 the testsuite:

```
$ make test-all
```

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.

4.3 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 and 3.4. Check https://travis-ci.org/elbenfreund/hamster-dbus/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To have a quick and dirty run of your tests without using tox and additional linters etc. . . :

```
$ make test
```

CHAPTER 5

Credits

5.1 Development Lead

- Eric Goller <eric.goller@ninjaduck.solutions>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

Packaging

hamster-bus follows the semantic versioning scheme. Each release is packaged and uploaded to pypi. We provide a compliant setup.py which contains all the meta information relevant to users of hamster-dbus. If you stumble upon any incompatibilities or dependency issue please let us know. If you are interested in packaging hamster-dbus for your preferred distribution or in some other context we would love to hear from you!

6.1 About requirements/*.pip

We do fully follow Donald Stuffts argument that information given setup.py is of fundamentally different nature than what may be located under requirements/*.pip (Additional comments can be found in the packaging guide and with Hynek Schlawack). As far as packaging goes setup.py is authoritative. We provide a set of specific environments under requirements/* that mainly developers and 3rd parties may find useful. This way we can easily enable contributers to get a suitable virtualenv running or specify our test environment in one central location. If for example you wanted to package hamster-dbus for debian-stable, it would be mighty convenient to just provide another requirements.txt with all the relevant dependencies pinned to what your target distro would provide. Now you can run the entire test suit against a reliable representation of said target system.

CHAPTER 7

History

CHAPTER 8

0.10.0 (2016-04-26)

- First release on PyPI.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search