Database Tools Documentation

Release 0.0.2

Gus Dunn

Jan 09, 2018

Contents

Contents:

CHAPTER 1

Database Tools

A set of command line executable and script importable tools to aid the Snapper Lab in managing and combining RedCap, FreezerPro, and other databases.

- Free software: MIT license
- Documentation: https://db-tools.readthedocs.io.

1.1 Features

- TODO

1.2 Install for Development

1. Install and become familiar with conda/Anaconda.
2. Fork the repository to your github by clicking here.
3. Clone your forked repo to your dev computer: `git clone git@github.com:YOUR_GITHUB_NAME/db_tools.git`.
4. Enter your freshly cloned Database Tools directory: `cd db_tools`.
5. Run `make help` to see most of the `make` targets available.
6. Running `make install`. This creates and registers a `conda` environment named db_tools. Into that conda environment, it installs all of the needed libraries to run and develop Database Tools.
7. To uninstall your dev environment just run `make uninstall-conda-env`. All traces of the environment should be erased.
8. Remember to activate the `conda` env before you try to use or interact with Database Tools or you will not have access to it.

1.3 Credits

This package was created with Cookiecutter and the xguse/cookiecutter-pypackage project template which is based on audreyr/cookiecutter-pypackage.

CHAPTER 2

Installation

2.1 Stable release

To install Database Tools, run this command in your terminal:

```
$ pip install db_tools
```

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

You can either clone the public repository:

```
$ git clone git://github.com/xguse/db_tools
```

Or download the tarball:

```
$ curl  -OL https://github.com/xguse/db_tools/tarball/master
```

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

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

CHAPTER 3

Usage

To use Database Tools in a project:

```
import db_tools
```

CHAPTER 4

Source Code Documentation

4.1 db_tools package

4.1.1 Submodules

4.1.2 db_tools.cli module

Console script for db_tools.

4.1.3 db_tools.db_tools module

Main module.

4.1.4 Module contents

Top-level package for Database Tools.

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/xguse/db_tools/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

Database Tools could always use more documentation, whether as part of the official Database Tools 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/xguse/db_tools/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 *db_tools* for local development.

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

```
$ git clone git@github.com:your_name_here/db_tools.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 db_tools
$ cd db_tools/
$ 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 db_tools 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.

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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/xguse/db_tools/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_db_tools
```

CHAPTER 6

Credits

6.1 Development Lead

- Gus Dunn <w.gus.dunn@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 v0.0.2 / 2018-01-09

- README: fixed urls in dev install docs

7.2 v0.0.1 / 2018-01-09

- First commit

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

D