tvrage3 Documentation

Release 0.1.0

Kalle Lindqvist

May 08, 2014

Contents

Contents:

CHAPTER 1

tvrage3

Python3 client for accessing tv show information from www.tvrage.com

- Free software: BSD license
- Documentation: http://tvrage3.rtfd.org.

1.1 Features

- Lazy, you can search tvrage with quick-search and still get all the information as you would get with a full search about the specified show. When a Show object is asked to return information not provided by the search method used, it will query tvrage for the information.
- Will handle the occasional database errors and information inconsistencies in the tvrage database sane and gracefully.
- High-level api, handles all the XML stuff for you.

1.2 Usage

- Searching
  - Full search

    Returns a list of Show objects.

    ```
    from tvrage3.search import search
    results = search('Buffy')
    first = results[0]
    first.name # => 'Buffy the Vampire Slayer'
    ```

  - Quick search

    Returns a show object, the closest match to search term or None.

    ```
    from tvrage3.search import quick_info
    result = quick_info('Csi crime')
    result.name # => 'CSI: Crime Scene Investigation'

    # Enable stricter matching
    result = quick_info('CSI crime', exact=True)
    result == None # => True
    ```

  - Search by id

    Returns a Show object, or None if id is incorrect.

    ```
    from tvrage3.search import search_id
    result = search_id('2930')
    result.name # => 'Buffy the Vampire Slayer'
    ```

- Show objects

  Show objects should not be initialized manually, it should be done by one of the search functions, but for this example we do.

  ```
  from tvrage3.api import Show
  show = Show(show_id='3183')

  show.air_day          # => 'Wednesday'
  show.air_time         # => '22:00'
  show.classification   # => 'Scripted'
  show.country          # => 'US'
  show.ended_year       # => None
  show.genres           # => ['Action', 'Crime', 'Drama']
  show.link             # => 'http://www.tvrage.com/CSI'
  show.name             # => 'CSI: Crime Scene Investigation'
  show.network          # => OrderedDict([('@country', 'US'), ('#text', 'CBS')])
  show.runtime          # => 60
  show.seasons          # => 14
  show.show_id          # => '3183'
  show.started_year     # => 2000
  show.status           # => 'Returning Series'
  ```

CHAPTER 2

Installation

At the command line:

```
$ easy_install tvrage3
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv tvrage3
$ pip install tvrage3
```

CHAPTER 3

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:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/kalind/tvrage3/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.

3.1.2 Fix Bugs

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

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

3.1.4 Write Documentation

tvrage3 could always use more documentation, whether as part of the official tvrage3 docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/kalind/tvrage3/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 :)

3.2 Get Started!

Ready to contribute? Here's how to set up *tvrage3* for local development.

1. Fork the *tvrage3* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/tvrage3.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 tvrage3
   $ cd tvrage3/
   $ 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 tvrage3 tests
   $ python setup.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.

3.3 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, and 3.3, and for PyPy. Check https://travis-ci.org/kalind/tvrage3/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_tvrage3
```

CHAPTER 4

Credits

4.1 Development Lead

- Kalle Lindqvist <kalle.lindqvist@mykolab.com>

4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

History

5.1 0.1.0 (2014-05-08)

- First release on PyPI.

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*