ramapi Documentation

Release 0.1.0

Rohan Hazra

Feb 23, 2022

Contents:

CHAPTER 1

ramapi

Python implementation for the Rick and Morty API https://rickandmortyapi.com/

Only works for Python3

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

1.1 Features

All methods returns json

Base class features:

- api_info() : api information
- schema() : json outline

Character,Location,Episode class features:

- get_all() : All information in paginated way
- get() : Information regarding the passed parameter
- filter() : Filtered results
- schema() : json outline

For detailed information and usage instructions:

- Read Docs at https://ramapi.readthedocs.io
- Visit official API Docs https://rickandmortyapi.com/documentation

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install ramapi, run this command in your terminal:

```
$ pip install ramapi
```

This is the preferred method to install ramapi, 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. [As this project is on Python3, pip3 is required]

2.2 From sources

The sources for ramapi can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/curiousrohan/ramapi
```

Or download the tarball:

```
$ curl  -OL https://github.com/curiousrohan/ramapi/tarball/master
```

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

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

CHAPTER 3

Usage

To use ramapi in a project:

```
import ramapi
```

This will import the ramapi package.

To use the Base class, Character Class or any other class:

```
from ramapi import Base
from ramapi import Character
```

To import all the classes:

```
from ramapi import *
```

Assuming you have imported the module successfully..

Base Class commands:

```
ramapi.Base.api_info()
ramapi.Base.schema()
```

Character/Location/Episode Class commands:

```
ramapi.Character.get_all()
ramapi.Character.get_page() //Only available for character class

ramapi.Character.get()
ramapi.Character.filter()
ramapi.Character.schema()
```

Replace Character with Location,Episode to access the corresponding properties.

All methods return json.

1. get_all()

This method doesn't take any parameters and returns all the resulsts in a paginated way.

Example:

```
ramapi.Episode.get_all()
```

2. get_page()

Takes page number as parameter and returns response of that page.

Example:

```
ramapi.Character.get_page(3)
```

3. get()

Take one or multiple parametes and returns corresponding output

Example:

```
ramapi.Location(4)
ramapi.Episode([10,28]) //Takes list as parameter
```

4. filter()

Takes one or more arguments and filters the results.

Example:

```
ramapi.Character.filter(name='rick',status='alive')
```

5. schema()

Returns the json outline for an particular class.

Example:

```
ramapi.Location.schema()
```

Checkout official docs at https://rickandmortyapi.com/documentation for more information.

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://github.com/curiousrohan/ramapi/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 GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

ramapi could always use more documentation, whether as part of the official ramapi 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://github.com/curiousrohan/ramapi/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 *ramapi* for local development.

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

```
$ git clone git@github.com:your_name_here/ramapi.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 ramapi
$ cd ramapi/
$ 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 the tests, including testing other Python versions with tox:

```
$ 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.

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 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/curiousrohan/ramapi/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Rohan Hazra <rohanhazra4@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2018-08-08)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search