swapi-python Documentation

Release 0.1.0

Paul Hallett

June 21, 2016

Contents

Contents:

CHAPTER 1

swapi-python

A Python helper library for swapi.co - the Star Wars API

NOTE: Tests will run against hosted API as opposed to data from github repo

- Free software: BSD license
- Documentation: https://swapi-python.readthedocs.org.

CHAPTER 2

Installation

At the command line:

```
$ pip install swapi
```

2.1 Basic Usage

To use swapi-python in a project:

```
import swapi
```

All resources are accessible through the top-level `get_resource()` methods:

```
luke = swapi.get_person(1)
tatooine = swapi.get_planet(1)
```

2.2 Methods

These are the top-level methods you can use to get resources from swapi.co. To learn more about the models and objects that are returned, see the `models` page.

2.2.1 get_person(id)

Return a single `Person` resource.

Example:

```
swapi.get_person(1)
>>> <Person - Luke Skywalker>
```

2.2.2 get_planet(id)

Return a single `Planet` resource.

Example:

```
swapi.get_planet(1)
>>> <Planet - Tatooine>
```

2.2.3 get_starship(id)

Return a single `Starship` resource.

Example:

```
swapi.get_starship(6)
>>> <Starship - Death Star>
```

2.2.4 get_vehicle(id)

Return a single `Vehicle` resource.

Example:

```
swapi.get_vehicle(4)
>>> <Vehicle - Sand Crawler>
```

2.2.5 get_film(id)

Return a single `Film` resource.

Example:

```
swapi.get_film(1)
>>> <Film - A New Hope>
```

2.2.6 get_all("resource")

Return a `QuerySet` containing all the items in a single resource. See the `'models'` page for more information on the models used in swapi-python.

Example:

```
swapi.get_all("films")
>>> <FilmQuerySet - 6>
```

CHAPTER 3

Models

Swapi-python uses custom models to handle single resources and collections of resources in a native and pythonic way.

This page documents the extra methods and functions of those classes.

3.1 Single Resource Model

A single resource model maps all the standard attributes for a resource as properties on the object. For example, a `People` resource has a `name` and `height` attribute. These can be accessed with swapi-python as properties, like so:

```
luke = swapi.get_person(1)
luke.name
>>> u'Luke Skywalker'
luke.height
>>> u'172'
```

3.1.1 Getting Related Resources

Each resource model has specific methods for getting back the collection of resources related to it. For example, the People resource can have Starship Resources linked to it. With swapi-python you can get those resources like so:

```
luke = swapi.get_person(1)
ships = luke.get_starships()
```

Each resource model has slightly different methods. Multiple resources return a Multiple Collection Model (see below).

3.1.2 People

- `get_starships()`

Return the starships that this person has piloted.

- `get_vehicles()`

Return the vehicles that this person has piloted.

- `get_homeworld()`

Get the homeworld of this person.

- `get_species()`

Return the species of this person.

- `get_films()`

Return the films that this person has been in.

3.1.3 Species

- `get_films()`

Return the films that this species has been in.

- `get_people()`

Return the people who are a member of this species.

- `get_homeworld()`

Return the homeworld of this species.

3.1.4 Planet

- `get_residents()`

Return the people who live on this planet.

- `get_films()`

Return the films that this planet has appeared in.

3.1.5 Starship

- `get_pilots()`

Return the pilots of this starship.

- `get_films()`

Return the films that this starship has been in.

3.1.6 Vehicle

- `get_pilots()`

Return the pilots of this vehicle.

- `get_films()`

Return the films that this vehicle has been in.

3.1.7 Film

- `get_starships()`

Get the starships in this film.

- `get_characters()`

Get the characters in this film.

- `get_vehicles()`

Get the vehicles in this film.

- `get_planets()`

Get the planets in this film.

- `get_species()`

Get the species in this film.

- `gen_opening_crawl()`

A generator yielding each line of the opening crawl for this film.

- `print_crawl()`

A novelty method that prints out each line of the opening crawl with a 0.5 second delay between each line.

3.2 Multiple Collection Model

When you query swapi.co for multiple resources of the same type, they will be returned as a `ResourceQuerySet`, which is a collection of those resources that you requested. For example, to get the `Starship` resources linked to a person, you can do the following:

```
luke = swapi.get_person(1)
starships = luke.get_starships()
>>> <StarshipQuerySet - 2>
```

`ResourceQuerySet` models have additional methods for dealing with multiple resources.

The items are accessible as the `item` property on the ResourceQuerySet if you want to directly access them:

```
starships.items
>>> [<Starship - X-wing>, <Starship - Lambda Shuttle>]
```

3.2.1 .order_by("attribute")

Return the list of Single Resource Models in this collection, ordered by a particular attribute. For example:

```
planets = swapi.get_all("planets")
for p in planets.order_by("diameter"):
    print(p.name)
```

This will return the planets ordered by their diameter.

This method will try to turn the attribute into an interger before ordering them, as most resources are unicode strings. If you try to order by string, it will order it alphabetically. Please be aware that for string ordering it might not always come out as you would expect.

3.2.2 .count()

Return a count of the number of items in this collection..

3.2.3 .iter()

An iterable method that can be used to iterate over each item in this collection:

```
for v in vehicles.iter():
    print(v.name)
```

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/phalt/swapi-python/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" is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

swapi-python could always use more documentation, whether as part of the official swapi-python 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/phalt/swapi-python/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 *swapi-python* for local development.

1. Fork the *swapi-python* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/swapi-python.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 swapi-python
   $ cd swapi-python/
   $ 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 swapi-python 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.

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 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/phalt/swapi-python/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_swapi-python
```

CHAPTER 5

Credits

5.1 Development Lead

- Paul Hallett <paulandrewhallett@gmail.com>

5.2 Contributors

Colm Harrington <colm.harrington@gmail.com>

CHAPTER 6

History

CHAPTER 7

0.1.0 (2014-12-21)

- First release on PyPI.

CHAPTER 8

0.1.2 (2014-12-22)

- Python 3.3 and Python 3.4 compatability

CHAPTER 9

0.1.3 (2014-12-22)

- Add “swapi-python” to User-Agent to help with analytics

CHAPTER 10

Indices and tables

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