selectorlib Documentation

Release 0.16.0

scrapehero

Jan 08, 2020

Contents:

CHAPTER 1

selectorlib

A library to read a YML file with Xpath or CSS Selectors and extract data from HTML pages using them

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

1.1 Example

```
>>> from selectorlib import Extractor
>>> yaml_string = """
    title:
        css: "h1"
        type: Text
    link:
        css: "h2 a"
        type: Link
    """
>>> extractor = Extractor.from_yaml_string(yaml_string)
>>> html = """
    <h1>Title</h1>
    <h2>Usage
        <a class="headerlink" href="http://test">¶</a>
    </h2>
    """
>>> extractor.extract(html)
{'title': 'Title', 'link': 'http://test'}
```

1.1.1 Installation

Stable release

To install selectorlib, run this command in your terminal:

```
$ pip install selectorlib
```

This is the preferred method to install selectorlib, 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.

From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/scrapehero/selectorlib
```

Or download the tarball:

```
$ curl -OL https://github.com/scrapehero/selectorlib/tarball/master
```

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

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

Selectorlib lets you use a YML styled file to specify the selectors for the elements or data that you need to extract from a website. You can use both CSS Selectors, XPaths or both.

1.1.2 YML Structure

Lets take a look at this fictional store that sells Pokemon - https://scrapeme.live/shop/

Lets extract Here is a sample YML that SelectorLib accepts as Input

```
pokemon:
    css: li.product
    multiple: true
    type: Text
    children:
        name:
            css: h2.woocommerce-loop-product__title
            type: Text
        price:
            css: span.woocommerce-Price-amount
            type: Text
        image:
            css: img.attachment-woocommerce_thumbnail
            type: Attribute
            attribute: src
        url:
            css: a.woocommerce-LoopProduct-link
            type: Link
```

Here `pokemon` is the main element and the elements - name, price, image and url are inside it and are called the children of the pokemon element.

Every element starts with its name and can have these properties

- css
- xpath
- type
- children
- formatter

css (default: Blank)

The css selector for the element. In our example the element called pokemon is in an li with a class product. So its `li.product`.

xpath (default: Blank)

The xpath selector for the element. If we were to use xpaths instead of css selectors for the element pokemon above. It would be `//li[contains(@class,'pokemon')]`. Every element needs either css or xpath selectors.

Every element needs either css or xpath selectors. If both xpath and css are defined, xpath takes preference.

type (default: Text)

The type defines what kind of extraction needs to happen on the selected element. Here are accepted types

Text

This type of extraction just extracts all the text content from the selected elements. If you have not specifed a type, Text would be used as default.

Attribute

This type of extraction lets you extract a particular attribute, specified using the `attribute` property for the element. This is not usually required when you are selecting using xpaths as you define that easily in an expression as compared to css selectors. eg. `//img[@src]`

Here is an example that extracts the src attribute of an img element

```
image:
    css: img.attachment-woocommerce_thumbnail
    type: Attribute
    attribute: src
```

Link

This type is a shortcut for getting the href attribute from any links in the html defined using an <a> tag

Example,

```
url:
    css: a.woocommerce-LoopProduct-link
    type: Link
```

HTML

HTML type, just gives you the full HTML content of the element. This is useful when you need the html as is for some custom extraction or checking a few conditions.

multiple (default: False)

If you need multiple matches on the selector of an element use multiple as true. If you only need to get the first match, use multiple as false or leave it blank. For example, the element pokemon has multiple matches on the same page, so we have set multiple:true in it to get all of them.

children (default: Blank)

An element can have multiple child elements. In the example above the parent element `pokemon` has these "children" - `name`,`price`,`image`,`url`. Each child element could also more children and can be nested. If an element has children, it's `type` property is ignored.

format

You can define custom formatters, and can be used for minor transformations on the extracted data. In Python, these formatters are defined as

```
from selectorlib.formatter import Formatter

class Price(Formatter):
    def format(self, text):
        return text.replace('\\n','').strip()
```

Used in the YAML as

```
price:
    css: span.woocommerce-Price-amount
    type: Text
    format: Price
```

And passed to the Extractor while its initialized

```
formatters = Formatter.get_all()
Extractor.from_yaml_file('a.yaml', formatters=formatters)
```

1.1.3 Python Example

scrapeme_listing_page.yml

```
pokemon:
    css: li.product
    multiple: true
    type: Text
    children:
        name:
            css: h2.woocommerce-loop-product__title
            type: Text
        price:
            css: span.woocommerce-Price-amount
            type: Text
        image:
            css: img.attachment-woocommerce_thumbnail
            type: Attribute
            attribute: src
        url:
            css: a.woocommerce-LoopProduct-link
            type: Link
```

extract.py

```
import requests
from selectorlib import Extractor, Formatter
from pprint import pprint
import re

# Define a formatter for Price
class Price(Formatter):
    def format(self, text):
        price = re.findall(r'\d+\.\d+',text)
        if price:
            return price[0]
        return None
formatters = Formatter.get_all()
extractor = Extractor.from_yaml_file('./scrapeme_listing_page.yml',
↪formatters=formatters)

#Download the HTML and use Extractor
r = requests.get('https://scrapeme.live/shop/')
data = extractor.extract(r.text)
pprint(data)
```

```
>>> python extract.py
```

```
{'pokemon': [{'image': 'https://scrapeme.live/wp-content/uploads/2018/08/001-350x350.
↪png',
              'name': 'Bulbasaur',
              'price': '63.00',
              'url': 'https://scrapeme.live/shop/Bulbasaur/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/002-350x350.
↪png',
              'name': 'Ivysaur',
              'price': '87.00',
```

(continues on next page)

(continued from previous page)

```
              'url': 'https://scrapeme.live/shop/Ivysaur/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/003-350x350.
↪png',
              'name': 'Venusaur',
              'price': '105.00',
              'url': 'https://scrapeme.live/shop/Venusaur/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/004-350x350.
↪png',
              'name': 'Charmander',
              'price': '48.00',
              'url': 'https://scrapeme.live/shop/Charmander/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/005-350x350.
↪png',
              'name': 'Charmeleon',
              'price': '165.00',
              'url': 'https://scrapeme.live/shop/Charmeleon/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/006-350x350.
↪png',
              'name': 'Charizard',
              'price': '156.00',
              'url': 'https://scrapeme.live/shop/Charizard/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/007-350x350.
↪png',
              'name': 'Squirtle',
              'price': '130.00',
              'url': 'https://scrapeme.live/shop/Squirtle/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/008-350x350.
↪png',
              'name': 'Wartortle',
              'price': '123.00',
              'url': 'https://scrapeme.live/shop/Wartortle/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/009-350x350.
↪png',
              'name': 'Blastoise',
              'price': '76.00',
              'url': 'https://scrapeme.live/shop/Blastoise/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/010-350x350.
↪png',
              'name': 'Caterpie',
              'price': '73.00',
              'url': 'https://scrapeme.live/shop/Caterpie/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/011-350x350.
↪png',
              'name': 'Metapod',
              'price': '148.00',
              'url': 'https://scrapeme.live/shop/Kakuna/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/015-350x350.
↪png',
              'name': 'Beedrill',
              'price': '168.00',
              'url': 'https://scrapeme.live/shop/Beedrill/'},
             {'image': 'https://scrapeme.live/wp-content/uploads/2018/08/016-350x350.
↪png',
              'name': 'Pidgey',
              'price': '159.00',
              'url': 'https://scrapeme.live/shop/Pidgey/'}]}
```

1.1.4 API Reference

Extractor Class

class selectorlib.selectorlib.**Extractor**(*config*, *formatters=None*)
selector class

extract(*html: str*, *base_url: str = None*)

Args: html: html string base_url (str, optional): specifying the base_url will make all extracted Links absolute

Returns: dict: extracted data from given html string

```
>>> response = requests.get(url)
>>> extractor.extract(response.text, base_url=response.url)
```

classmethod from_yaml_file(*yaml_filename: str*, *formatters=None*)
create *Extractor* object from yaml file

```
>>> extractor = Extractor.from_yaml_string('selectors.yaml')
```

classmethod from_yaml_string(*yaml_string: str*, *formatters=None*)
create *Extractor* object from yaml string

```
>>> yaml_string = '''
    title:
        css: "h1"
        type: Text
    '''
>>> extractor = Extractor.from_yaml_string(yaml_string)
```

Formatter Class

class selectorlib.formatter.**Formatter**
Inherit this class and override format function

format(*text: str*)
Override this function in inherited subclass. return text after formatting

classmethod get_all()
returns all subclasses inherited from Formatter

```
>>> formatters = Formatter.get_all()
>>> Extractor.from_yaml_file('a.yaml', formatters=formatters)
```

1.1.5 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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/scrapehero/selectorlib/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

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

Get Started!

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

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

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

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

Tips

To run a subset of tests:

```
$ py.test tests.test_selectorlib
```

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.

1.1.6 Credits

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

Development Lead

- Ashwin Rajeev <ashwin@scrapehero.com>

Contributors

None yet. Why not be the first?

1.1.7 History

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index