PyScrapper

Release v1.0.0

Feb 25, 2020

Contents

PyScrapper is a web scrapping tool. It helps to scrape webpages and form a meaningful json object, as per the given configuration. Configuration is what tells the scrapper, which blocks of the html needs to be parsed and how they should be structurized for ease of use.

CHAPTER 1

Table of Contents

1.1 User Guide

1.1.1 Installing PyScrapper

The preferred installation method is by using pip:

```
$ pip install pyscrapper
```

If you don't have pip installed, you can easily install it by downloading and running get-pip.py.

If, for some reason, pip won't work, you can manually download the PyScrapper distribution from PyPI, extract and then install it:

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

1.1.2 Code examples

The source distribution contains the `examples` directory where you can find many working examples for using PyScrapper in different ways. The examples can also be browsed online.

1.1.3 Basic Concepts

PyScrapper has an assembly module which is an assembly of base modules:

- urlloaders
- observers
- managers

UrlLoaders are different types of url request makers such as a Web Browser, a simple GET request etc., anything which is capable of loading an url and give the resultant html

Observers are those elements, which are triggered when an url loading is completed and scrapping is completed.

Managers manage the load balancing tasks and restrict the number requests executing parallely as per the user's configuration. They load the url in an `urlloader`, once the result is received from the `observer` then parse the html content using scrapping module, and again pushes back the final scrapped result to a user registered observer.

Now you might have got an abstract idea, on what each module of assembly does. Let's have deep dive into each of these modules.

1.1.4 Observers

`pyscrapper.assembly.observers`

The observers module has two interfaces

- **Observable :**
  - An Observable can hold a list of observers.
  - When an event occurs, such as url loaded / scraping completed , then all the observers are notified with the change.
- **Observer :**
  - An Observer is capable of listening to a change and trigger the corresponding operations.

1.1.5 UrlLoaders

`pyscrapper.assembly.urlloaders`

- The UrlLoader interface inherits the Observable interface
- Any class implementing the UrlLoader interface is capable of holding and notifying a list of observers.

1.1.6 Managers

`pyscrapper.assembly.managers`

- Manager acts both, as Observable to the user requests and Observer to the UrlLoader
- As an Observable, the manager holds all the Observers, defined by the user / developer, who are interested in receiving data when published.
- As an Observer, the manager becomes is registered with UrlLoader(which is also an Observable), to get notified on url response is received.

1.1.7 Scrapper Config

The configuration is the major part, which tells the scrapper, which blocks of the html needs to be parsed and how they should be structurized for ease of use.

- Building blocks of configuration.
  - `listItem (string)`: The list item selector.
  - `data (string | object)`: The fields to include in the list objects:

- `<fieldName>(string | object)`: The selector or an object containing:
    - `selector (string)`: The selector.
    - `attr (string)`: If provided, the value will be taken based on the attribute name.
    - `eq (int)`: If provided, it will select the nth element.
    - `function(a callable function)`: If provided, it will be called with the current block's data, obtained after parsing the html of current block, on which the user can perform any operation and must return a result… which is then considered as final result of that block.

Warning: (listItem, data, selector, attr, eq, function) all of these are keywords. Do not ever try to use any of the listed keywords as your field names, because it may conflict the process of parsing.

Check out the examples , to get better understanding.

1.2 Version History

1.3 API Reference