Scraper Toolkit Documentation

Release 1.0.0

Anthony Michael Pruitt

Jun 08, 2019

Classes:

A toolkit to assist in the page-fetching, HTML-parsing, and data-exporting of a web scraping project.

CHAPTER 1

ScraperProject

class scraper_toolkit.ScraperProject.**ScraperProject**(*domain: str*)

Handle the page fetching, HTML parsing, and exporting of a web scraping project.

Parameters **domain** – Prefix to be added to scraped URLs missing the domain.

add_selector(*selector: Union[str, Selector], attribute: str = None, name: str = None, post_processing: Callable = None*)

Add the given selector to loaded CSS selectors.

Parameters

- **selector** – CSS selector as a string or a Selector type object.
- **attribute** – HTML attribute of the element to store
- **name** – Optional name for the parsed attribute, useful for creating the header row when exporting as a CSV file.
- **post_processing** – Optional function called on the parsed attribute before it is stored. Useful for cleaning up and splitting data.

add_selectors(*selectors: List[Selector]*)

Add multiple CSS selectors to loaded selectors.

Parameters **selectors** – List of Selector objects.

export_to_csv(*csv_path: pathlib.Path*, *encoding: str = 'UTF-8'*, *write_header: bool = True*)

Export parsed data to a CSV file.

Parameters

- **csv_path** – Path of the location to save the CSV file.
- **encoding** – CSV file encoding. Default is UTF-8.
- **write_header** – If true, write a header row to the CSV file using the "name" keys in the provided data.

fetch(*url: str = None*) → str

Fetch HTML from the page at the given URL.

Parameters **url** – URL of the target page.

Returns HTML page as a string.

parse()

Parse HTML for elements using loaded CSS selectors and append matching elements to self.parsed as dictionary objects.

CHAPTER 2

PageFetcher

class scraper_toolkit.components.PageFetcher.**PageFetcher**(*domain: str*)

Fetch URLs and return web pages' HTML as strings.

Parameters **domain** – Prefix to be added to scraped URLs missing the domain.

static get_full_url(*domain: str*, *suffix: str*) → str

Return a complete URL given a domain and suffix, even if the provided suffix is the complete URL.

Parameters

- **domain** – The domain of the target page URL.
- **suffix** – The URL of the target page, with or without the domain prefix.

Returns The complete URL.

get_html(*url: str = None*) → str

Fetch the page HTML from the given URL.

Parameters **url** – URL of target page.

Returns HTML as a string.

get_links_from_page(*target_url: str = None*) → Iterable[str]

Return a list of every href URL found from target_url.

Parameters **target_url** – URL of page to search for href links.

Returns List of every discovered href link on the page.

static select_elements_from_html(*html: str*, *selector: str*)

Return a list of HTML elements from the given html that match the provided CSS selector.

Parameters

- **html** – HTML of the page to parse.
- **selector** – CSS selector for target elements.

Returns List of HTML elements matching the CSS selector.

select_links_from_page(*selector: str*, *target_url: str = None*) → Iterable[str]
Yield the HTML of all pages linked on the target_url located by the given CSS selector.

Parameters

- **selector** – CSS selector for elements containing href attribute
- **target_url** – URL to search for links. If none is provided, the domain URL will be used.

Returns Generator for HTML as strings, fetched from the selected links.

CHAPTER 3

Parser

class scraper_toolkit.components.Parser.**Parser**(*html: str*)

Parse HTML for specific elements or attributes

Parameters **html** – HTML to parse, as a string.

add_selector(*selector: Union[str, scraper_toolkit.components.Selector.Selector] = None*, *attribute: str = None*, *name: str = None*, *post_processing: Callable = None*)

Add the given selector to loaded CSS selectors.

Parameters

- **selector** – CSS selector as a string or a Selector type object.
- **attribute** – HTML attribute of the element to store
- **name** – Optional name for the parsed attribute, useful for creating the header row when exporting as a CSV file.
- **post_processing** – Optional function called on the parsed attribute before it is stored. Useful for cleaning up and splitting data.

parse()

Parse HTML for elements using loaded CSS selectors and append matching elements to self.parsed as dictionary objects.

CHAPTER 4

Exporter

class scraper_toolkit.components.Exporter.**Exporter**(*data: Union[Parser, dict, List[dict]]*)

Export data from parsers.

Param data: The data to export as a Parser object, a dictionary, or a list of dictionaries.

export_to_csv(*csv_path: Union[pathlib.Path, pathlib.PurePath, str], encoding: str = 'UTF-8', write_header: bool = True*)

Export parsed data to a CSV file.

Parameters

- **csv_path** – Path of the location to save the CSV file.
- **encoding** – CSV file encoding. Default is UTF-8.
- **write_header** – If true, write a header row to the CSV file using the "name" keys in the provided data.

CHAPTER 5

Selector

class scraper_toolkit.components.Selector.**Selector**(*selector_str: str*, *name: str = None*, *attribute: str = None*, *post_processing: Callable = None*)

Represent a CSS selector with an optional name and target attribute, with an optional post-processing function.

Parameters

- **selector_str** – CSS selector as a string.
- **name** – Optional name for the parsed attribute, useful for creating the header row when exporting as a CSV file.
- **attribute** – HTML attribute of the element to store
- **post_processing** – Optional function called on the parsed attribute before it is stored. Useful for cleaning up and splitting data.

Python Module Index

S

Index