Chopper Documentation

Release 0.3.0

Jurismarchés

July 25, 2014

Contents

Chopper is a tool to extract elements from HTML by preserving ancestors and CSS rules.

Contents:

CHAPTER 1

Installation

Chopper installation is super simple, no configuration at all.

1.1 Using pip

Simply run :

```
pip install chopper
```

1.2 Using setup.py

Download the source and run :

```
python setup.py install
```

CHAPTER 2

Usage

2.1 Create the `Extractor` instance

First, you need to import the `Extractor` class :

```
from chopper import Extractor
```

Then you can create an `Extractor` instance by explicitly instantiating one or by directly using `Extractor.keep()` and `Extractor.discard()` class methods :

```
from chopper import Extractor

# Instantiate style
extractor = Extractor().keep('//div').discard('//a')

# Class method style
extractor = Extractor.keep('//div').discard('//a')
```

2.2 Add Xpath expressions

The `Extractor` instance allows you to chain multiple `Extractor.keep()` and `Extractor.discard()`

```
from chopper import Extractor

e = Extractor.keep('//div[p]').discard('//span').discard('//a').keep('strong')
```

2.3 Extract contents

Once your `Extractor` instance is created you can call the `Extractor.extract()` method on it. The `Extractor.extract()` method takes at least one argument that is the HTML to parse.

If you want to also parse CSS, pass it as the second argument.

```
from chopper import Extractor

HTML = """
<html>
  <head>
    <title>Hello world !</title>
```

```
  </head>
  <body>
    <header>This is the header</header>
    <div>
      <p><span>Main </span>content</p>
      <a href="/">See more</a>
    </div>
    <footer>This is the footer</footer>
  </body>
</html>
"""

CSS = """
a { color: blue; }
p { color: red; }
span { border: 1px solid red; }
body { background-color: green; }
"""

# Create the Extractor
e = Extractor.keep('//div[p]').discard('//span').discard('//a')

# Parse HTML only
html = e.extract(HTML)

>>> html
"""
<html>
  <body>
    <div>
      <p>content</p>
    </div>
  </body>
</html>
"""

# Parse HTML & CSS
html, css = e.extract(HTML, CSS)

>>> html
"""
<html>
  <body>
    <div>
      <p>content</p>
    </div>
  </body>
</html>
"""

>>> css
"""
p{color:red;}
body{background-color:green;}
"""
```

2.4 Convert relative links to absolute ones

Chopper can also convert relative links to absolute ones. To do so, simply use the *base_url* keyword arguments on the `Extractor.extract()` method.

```
from chopper import Extractor

HTML = """
<html>
  <head>
    <title>Hello world !</title>
  </head>
  <body>
    <div>
      <p>content</p>
      <a href="page.html">See more</a>
    </div>
  </body>
</html>
"""

html = Extractor.keep('//a').extract(HTML, base_url='http://test.com/path/index.html')

>>> html
"""
<html>
  <body>
    <div>
      <a href="http://test.com/path/page.html">See more</a>
    </div>
  </body>
</html>
"""
```

CHAPTER 3

API

3.1 *Extractor* public API

class `Extractor`

`keep` (*xpath*)
Adds an Xpath expression to keep

Parameters **xpath** (*str*) – The Xpath expression to add

Returns The self instance

Return type *Extractor*

`discard` (*xpath*)
Adds an Xpath expression to discard

Parameters **xpath** (*str*) – The Xpath expression to add

Returns The self instance

Return type *Extractor*

`extract` (*html_contents*, *css_contents=None*, *base_url=None*)
Extracts the cleaned html tree as a string and only css rules matching the cleaned html tree

Parameters

- **html_contents** (*str*) – The HTML contents to parse
- **css_contents** (*str*) – The CSS contents to parse
- **base_url** (*str*) – The base page URL to use for relative to absolute links

Returns cleaned HTML contents or (cleaned HTML contents, cleaned CSS contents)

Return type str or tuple

CHAPTER 4

Indices and tables

- *genindex*
- *search*