HTMLement Documentation

Release 2.0.0

William Forde

Apr 11, 2023

Contents

CHAPTER 1

HTMLement

HTMLement is a pure Python HTML Parser.

The object of this project is to be a “pure-python HTML parser” which is also “faster” than “beautifulsoup”. And like “beautifulsoup”, will also parse invalid html.

The most simple way to do this is to use ElementTree XPath expressions. Python does support a simple (read limited) XPath engine inside its “ElementTree” module. A benefit of using “ElementTree” is that it can use a “C implementation” whenever available.

This “HTML Parser” extends html.parser.HTMLParser to build a tree of ElementTree.Element instances.

CHAPTER 2

Install

Run

```
pip install htmlement
```

-or-

```
pip install git+https://github.com/willforde/python-htmlement.git
```

CHAPTER 3

Parsing HTML

Here I'll be using a sample "HTML document" that will be "parsed" using "htmlement":

```
html = """
<html>
  <head>
    <title>GitHub</title>
  </head>
  <body>
    <a href="https://github.com/marmelo">GitHub</a>
    <a href="https://github.com/marmelo/python-htmlparser">GitHub Project</a>
  </body>
</html>
"""

# Parse the document
import htmlement
root = htmlement.fromstring(html)
```

Root is an ElementTree.Element and supports the ElementTree API with XPath expressions. With this I'm easily able to get both the title and all anchors in the document.

```
# Get title
title = root.find("head/title").text
print("Parsing: %s" % title)

# Get all anchors
for a in root.iterfind(".//a"):
    print(a.get("href"))
```

And the output is as follows:

```
Parsing: GitHub
https://github.com/willforde
https://github.com/willforde/python-htmlement
```

CHAPTER 4

Parsing HTML with a filter

Here I'll be using a slightly more complex "HTML document" that will be "parsed" using "htmlement with a filter" to fetch only the menu items. This can be very useful when dealing with large "HTML documents" since it can be a lot faster to only "parse the required section" and to ignore everything else.

```
html = """
<html>
  <head>
    <title>Coffee shop</title>
  </head>
  <body>
    <ul class="menu">
      <li>Coffee</li>
      <li>Tea</li>
      <li>Milk</li>
    </ul>
    <ul class="extras">
      <li>Sugar</li>
      <li>Cream</li>
    </ul>
  </body>
</html>
"""

# Parse the document
import htmlement
root = htmlement.fromstring(html, "ul", attrs={"class": "menu"})
```

In this case I'm not unable to get the title, since all elements outside the filter were ignored. But this allows me to be able to extract all "list_item elements" within the menu list and nothing else.

```
# Get all listitems
for item in root.iterfind(".//li"):
    # Get text from listitem
    print(item.text)
```

And the output is as follows:

```
Coffee
Tea
Milk
```

See also:

More examples can be found in examples.py.

CHAPTER 5

API

5.1 HTMLement

Simple lightweight HTML parser with XPath support.

Github: https://github.com/willforde/python-htmlement Documentation: https://python-htmlement.readthedocs.io/en/stable/?badge=stable Testing: https://github.com/willforde/python-htmlement/actions Coverage: https://codecov.io/gh/willforde/python-htmlement Maintainability: https://codeclimate.com/github/willforde/python-htmlement/maintainability

class htmlement.**HTMLement**(*tag=''*, *attrs=None*, *encoding=None*)

Python HTMLParser extension with ElementTree Parser support.

This HTML Parser extends `html.parser.HTMLParser`, returning an `xml.etree.ElementTree.Element` instance. The returned root element natively supports the ElementTree API. (e.g. you may use its limited support for XPath expressions)

When a "tag" and "tag attributes" are given the parser will search for a required section. Only when the required section is found, does the parser start parsing the "HTML document". The element that matches the search criteria will then become the new "root element".

Attributes are given as a dict of {'name': 'value'}. Value can be the string to match, *True* or *False*. *True* will match any attribute with given name and any value. *False* will only give a match if given attribute does not exist in the element.

Parameters

- **tag** (*str*) – (optional) Name of "tag / element" which is used to filter down "the tree" to a required section.
- **attrs** (*dict(str, str)*) – (optional) The attributes of the element, that will be used, when searchingfor the required section.
- **encoding** (*str*) – (optional) Encoding used, when decoding the source data before feeding it to the parser.

close()
Close the "tree builder" and return the "root element" of the "element tree".

Returns The "root element" of the "element tree".

Return type xml.etree.ElementTree.Element

Raises **RuntimeError** – If no element matching search criteria was found.

feed(*data*)
Feeds data to the parser.

If *data*, is of type `bytes` and where no encoding was specified, then the encoding will be extracted from *data* using "meta tags", if available. Otherwise encoding will default to "ISO-8859-1"

Parameters **data** (*str or bytes*) – HTML data

Raises **UnicodeDecodeError** – If decoding of *data* fails.

htmlement.**fromstring**(*text*, *tag=''*, *attrs=None*, *encoding=None*)
Parse's "HTML" document from a string into an element tree.

Parameters

- **text** (*str or bytes*) – The "HTML" document to parse.
- **tag** (*str*) – (optional) Name of "tag / element" which is used to filter down "the tree" to a required section.
- **attrs** (*dict(str, str)*) – (optional) The attributes of the element, that will be used, when searchingfor the required section.
- **encoding** (*str*) – (optional) Encoding used, when decoding the source data before feeding it to the parser.

Returns The root element of the element tree.

Return type xml.etree.ElementTree.Element

Raises **UnicodeDecodeError** – If decoding of *text* fails.

htmlement.**fromstringlist**(*sequence*, *tag=''*, *attrs=None*, *encoding=None*)
Parses an "HTML document" from a sequence of "HTML sections" into an element tree.

Parameters

- **sequence** (*list(str or bytes)*) – A sequence of "HTML sections" to parse.
- **tag** (*str*) – (optional) Name of "tag / element" which is used to filter down "the tree" to a required section.
- **attrs** (*dict(str, str)*) – (optional) The attributes of the element, that will be used, when searchingfor the required section.
- **encoding** (*str*) – (optional) Encoding used, when decoding the source data before feeding it to the parser.

Returns The root element of the element tree.

Return type xml.etree.ElementTree.Element

Raises **UnicodeDecodeError** – If decoding of a section within *sequence* fails.

htmlement.**parse**(*source*, *tag=''*, *attrs=None*, *encoding=None*)
Load an external "HTML document" into an element tree.

Parameters

- **source** (*str or io.TextIOBase*) – A filename or file like object containing HTML data.
- **tag** (*str*) – (optional) Name of "tag / element" which is used to filter down "the tree" to a required section.
- **attrs** (*dict(str, str)*) – (optional) The attributes of the element, that will be used, when searchingfor the required section.
- **encoding** (*str*) – (optional) Encoding used, when decoding the source data before feeding it to the parser.

Returns The root element of the element tree.

Return type xml.etree.ElementTree.Element

Raises **UnicodeDecodeError** – If decoding of *source* fails.

CHAPTER 6

External Links

ElementTree: https://docs.python.org/2/library/xml.etree.elementtree.html

Bug Tracker: https://github.com/willforde/python-htmlement/issues

Python Module Index

h

Index