ScraperWiki Classic has been retired.

Why not try our new Code in your browser tool? It's free and awesome!

Documentation / HTML parsing guide

Extract data from HTML using CSS selectors in Python

Back to contents | Choose a language: Ruby | Python | PHP

The easiest and most familiar way to extract data from HTML web pages is to use "CSS selectors". These are part of the same rules which in web stylesheets are used to describe the spacing, colour and layout of web pages.

For more details, read the lxml documentation, or the CSS selector specification.

Getting started

Grab the HTML web page, and parse the HTML using lxml.

```
import scraperwiki
import lxml.html
html = scraperwiki.scrape("https://scraperwiki.com/")
root = lxml.html.fromstring(html)
```

Select all **<a>** elements that are inside ** **. These queries work the same way as CSS stylesheets or jQuery. They are called CSS selectors, and are quite powerful.

```
for el in root.cssselect("div.featured a"):
    print el
```

Print out all of a tag and its contents as HTML.

```
    print lxml.html.tostring(el)
```

Read attributes, such as the target of the **<a>** tags.

```
    print el.attrib['href']
```

Simple text extraction

Select the first **** element inside ** **.

```
el = root.cssselect("div#footer_inner strong")[0]
print el
```

Extract the text from inside the tag.

```
print el.text
```

Any trailing text from just after the close tag.

```
print el.tail
```

Deep text extraction

Get all text recursively, throwing away any child tags.

```
eg = lxml.html.fromstring('<h2>A thing <b>goes boom</b> up <i>on <em>the tree</em></i
print eg.text_content() # 'A thing goes boom up on the tree'
```

Sometimes you have nearly pure text elements that still have <i> and <b> elements which you want to retain. Such an element can be extracted using a recursive function.

```
def ctext(el):
    result = [ ]
    if el.text:
        result.append(el.text)
    for sel in el:
        assert sel.tag in ["b", "i"]
        result.append("<"+sel.tag+">")
        result.append(ctext(sel))
        result.append("</"+sel.tag+">")
        if sel.tail:
            result.append(sel.tail)
    return "".join(result)
```

This gives an error if there are other unexpected elements, such as .

```
print ctext(eg)
```

Finding data manually

Iterate down through the elements in the document and see the tags and attributes on each element.

```
for el in root:
    print el.tag
    for el2 in el:
        print "--", el2.tag, el2.attrib
```

Navigate around the document.

```
eg = lxml.html.fromstring('<h2>A thing <b>goes boom</b> up <i>on <em>the tree</em></i
print eg[1].tag                 # i
print eg[1].getparent().tag     # h2
print eg[1].getprevious().tag   # b
print eg[1].getnext()           # None
print eg[1].getchildren()       # [<Element em>]
```