jsoup.org

Use selector-syntax to find elements: jsoup Java HTML parser

4-5 minutes

Problem

You want to find or manipulate elements using a CSS or jquery-like selector syntax.

Solution

Use the Element.select(String selector) and Elements.select(String selector) methods:

```
File input = new File("/tmp/input.html");
Document doc = Jsoup.parse(input, "UTF-8",
"http://example.com/");Elements links =
doc.select("a[href]"); // a with href
Elements pngs = doc.select("img[src$=.png]");
  // img with src ending .pngElement masthead
= doc.select("div.masthead").first();
  // div with class=mastheadElements
resultLinks = doc.select("h3.r > a"); //
```

```
direct a after h3
```

Description

jsoup elements support a CSS (or jquery) like selector syntax to find matching elements, that allows very powerful and robust queries.

The `select` method is available in a `Document`, `Element`, or in `Elements`. It is contextual, so you can filter by selecting from a specific element, or by chaining select calls.

Select returns a list of Elements (as `Elements`), which provides a range of methods to extract and manipulate the results.

Selector overview

- `tagname`: find elements by tag, e.g. `a`
- `ns|tag`: find elements by tag in a namespace, e.g. `fb|name` finds `<fb:name>` elements
- `#id`: find elements by ID, e.g. `#logo`
- `.class`: find elements by class name, e.g. `.masthead`
- `[attribute]`: elements with attribute, e.g. `[href]`
- `[^attr]`: elements with an attribute name prefix, e.g. `[^data-]` finds elements with HTML5 dataset attributes
- `[attr=value]`: elements with attribute value, e.g.

`[width=500]` (also quotable, like `[data-name='launch sequence']`)

- `[attr^=value]`, `[attr$=value]`, `[attr*=value]`: elements with attributes that start with, end with, or contain the value, e.g. `[href*=/path/]`
- `[attr~=regex]`: elements with attribute values that match the regular expression; e.g. `img[src~=(?i)\.(png|jpe?g)]`
- *: all elements, e.g. *

Selector combinations

- `el#id`: elements with ID, e.g. `div#logo`
- `el.class`: elements with class, e.g. `div.masthead`
- `el[attr]`: elements with attribute, e.g. `a[href]`
- Any combination, e.g. `a[href].highlight`
- `ancestor child`: child elements that descend from ancestor, e.g. `.body p` finds p elements anywhere under a block with class "body"
- `parent > child`: child elements that descend directly from parent, e.g. `div.content > p` finds p elements; and `body > *` finds the direct children of the body tag
- `siblingA + siblingB`: finds sibling B element immediately preceded by sibling A, e.g. `div.head + div`

- `siblingA ~ siblingX`: finds sibling X element preceded by sibling A, e.g. `h1 ~ p`
- `el, el, el`: group multiple selectors, find unique elements that match any of the selectors; e.g. `div.masthead, div.logo`

Pseudo selectors

- `:lt(n)`: find elements whose sibling index (i.e. its position in the DOM tree relative to its parent) is less than n; e.g. `td:lt(3)`
- `:gt(n)`: find elements whose sibling index is greater than n; e.g. `div p:gt(2)`
- `:eq(n)`: find elements whose sibling index is equal to n; e.g. `form input:eq(1)`
- `:has(seletor)`: find elements that contain elements matching the selector; e.g. `div:has(p)`
- `:not(selector)`: find elements that do not match the selector; e.g. `div:not(.logo)`
- `:contains(text)`: find elements that contain the given text. The search is case-insensitive; e.g. `p:contains(jsoup)`
- `:containsOwn(text)`: find elements that directly contain the given text
- `:matches(regex)`: find elements whose text matches the

specified regular expression; e.g. `div:matches((?i)login)`

- `:matchesOwn(regex)`: find elements whose own text matches the specified regular expression
- Note that the above indexed pseudo-selectors are 0-based, that is, the first element is at index 0, the second at 1, etc

  See the `Selector` API reference for the full supported list and details.