Scrapy Tutorial #8: Scrapy Selector Guide

Last updated on Feb 21 2019 by Michael Yin

Introduction:

Some readers might have questions when reading this scrapy tutorial series, what is the `extract` method, how to use it, what if I want to iterate a note list and extract the sub-nodes. In this post, I would talk about Scrapy Selector and how to use it with iteration.

Description

Scrapy have its own mechanism for extracting data which are called `selectors`, they can select the certain part of HTML by using XPath or CSS expression. XPath is designed to select info from XML document since Html is a special type of XML, so XPath can also be used to select info from HTML. CSS is designed to apply styles to HTML nodes, so it can also be used to select nodes from HTML. There is no solid answer to which exprssion is better, so just choose the one you like, in this scrapy tutorial for Python 3 I will use XPath with Scrapy Selector, you can also use CSS with Scrapy Selector.

Constructing Selectors

To make you understand about Scrapy Selector better, here we use real HTML source code to test.

```
<html>
   <head>
      <title>Scrapy Tutorial Series By MichaelYin</title>
   </head>
   <body>
      <div class='links'>
         <a href='one.html'>Link 1<img src='image1.jpg'/></a>
         <a href='two.html'>Link 2<img src='image2.jpg'/></a>
         <a href='three.html'>Link 3<img src='image3.jpg'/></a>
      </div>
   </body>
</html>
```

We can construct selector instance by passing `Text` or `TextResponse` object. First, let's enter Scrapy shell by using `scrapy shell`, then paste the code from blog post to the terminal. **What you should know here is that >>> indicates an interactive session and code typed in python shell are marked with this. Output is show without the arrows.**

```
>>> from scrapy.selector import Selector
>>> from scrapy.http import HtmlResponse
```

Now constructing selector from text

```
>>> body = """
<html>
   <head>
      <title>Scrapy Tutorial Series By MichaelYin</title>
   </head>
   <body>
```

```
      <div class='links'>
         <a href='one.html'>Link 1<img src='image1.jpg'/></a>
         <a href='two.html'>Link 2<img src='image2.jpg'/></a>
         <a href='three.html'>Link 3<img src='image3.jpg'/></a>
      </div>
   </body>
</html>
"""
>>> sel = Selector(text=body)
>>> sel.xpath("//title/text()").extract()
[u'Scrapy Tutorial Series By MichaelYin']
```

If you want to construct selector instance from response

```
>>> response = HtmlResponse(url="http://mysite.com", body=body, encoding='utf-8')
>>> Selector(response=response).xpath("//title/text()").extract()
[u'Scrapy Tutorial Series By MichaelYin']
```

Response object also exposed a selector on `selector` attribute to make it convenient to use the selector. So in most cases, you can directly use it.

```
>>> response.selector.xpath('//title/text()').extract()
>>> response.xpath('//title/text()').extract()
```

The code above has same outputs.

How to use Scrapy selectors

When you use `XPath` or `CSS` method to select nodes from HTML, the output returned by the methods is `SelectorList` instance.

```
>>> response.selector.xpath('//a/@href')
[<Selector xpath='//a/@href' data=u'one.html'>,
 <Selector xpath='//a/@href' data=u'two.html'>,
 <Selector xpath='//a/@href' data=u'three.html'>]
```

As you can see, SelectorList is a list of new selectors. If you want to get the textual data instead of `SelectorList`, just call `.extract` method.

```
>>> response.selector.xpath('//a/@href').extract()
[u'one.html', u'two.html', u'three.html']
```

You can use `extract_first` to extract only first matched element, which can save you from `extract()[0]`

```
>>> response.xpath("//a[@href='one.html']/img/@src").extract_first()
u'image1.jpg'
```

Nesting Selectors

Since `extract` returns a list of selectors, so we can also call `CSS` or `XPath` of these selectors to extract data. For example, we can first get the list of hyper link, then extract all text and image nodes.

```
>>> links = response.xpath("//a")
>>> links.extract()
[u'<a href="one.html">Link 1<img src="image1.jpg"></a>',
 u'<a href="two.html">Link 2<img src="image2.jpg"></a>',
 u'<a href="three.html">Link 3<img src="image3.jpg"></a>']
>>> for index, link in enumerate(links):
...     args = (index, link.xpath('@href').extract(), link.xpath('img/@src').extract())
...     print('Link number %d points to url %s and image %s' % args)
Link number 0 points to url ['one.html'] and image ['image1.jpg']
Link number 1 points to url ['two.html'] and image ['image2.jpg']
Link number 2 points to url ['three.html'] and image ['image3.jpg']
```

Conclusion

In this scrapy tutorial for Python 3, I talked about how to construct Scrapy selector, how to use it to extract data and how to use nesting selectors, all the code of this tutorial is Python 3, which is the future of Python. If you have any question about Scrapy, just leave me message here, I will respond ASAP.