XPath

XPath，全称 XML Path Language，即 XML 路径语言，它是一门在 XML 文档中查找信息的语言。最初是用来搜寻 XML 文档的，但同样适用于 HTML 文档的搜索。所以在做爬虫时完全可以使用 XPath 做相应的信息抽取。

1. XPath 概览

XPath 的选择功能十分强大，它提供了非常简洁明了的路径选择表达式。另外，它还提供了超过 100 个内建函数，用于字符串、数值、时间的匹配以及节点、序列的处理等，几乎所有想要定位的节点都可以用 XPath 来选择。　　官方文档：https://www.w3.org/TR/xpath/

2. XPath 常用规则

表达式	描述
nodename	选取此节点的所有子节点
/	从当前节点选区直接子节点
//	从当前节点选取子孙节点
.	选取当前节点
..	选取当前节点的父节点
@	选取属性

这里列出了 XPath 的常用匹配规则，示例如下：

```
//title[@lang='eng']
```

这是一个 XPath 规则，代表的是选择所有名称为 title，同时属性 lang 的值为 eng 的节点，后面会通过 Python 的 lxml 库，利用 XPath 进行 HTML 的解析。

3. 安装

> windows->python3环境下：pip install lxml
>
> linux环境下：pip3 install lxml

4. 实例引入

```
from lxml import etree

text = '''
<div>
<ul>
<li class="item-0"><a href="link1.html">first item</a></li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-inactive"><a href="link3.html">third item</a></li>
<li class="item-1"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a>
</ul>
</div>
'''
```

首先导入 lxml 库的 etree 模块，然后声明一段 HTML 文本，调用 HTML 类进行初始化，成功构造一个 XPath 解析对象。注意：HTML 文本中最后一个 li 节点没有闭合，但是 etree 模块可以自动修正 HTML 文本。　　调用 tostring() 方法即可输出修正后的 HTML 代码，但结果是 bytes 类型，可以用 decode() 方法将其转化为 str 类型，结果如下：

```
<html><body><div>
<ul>
<li class="item-0"><a href="link1.html">first item</a></li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-inactive"><a href="link3.html">third item</a></li>
<li class="item-1"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a>
</li></ul>
</div>
</body></html>
```

经过处理后，li 节点标签被补全，并且还自动添加了 body、html 节点。　　还可以直接读取文本文件进行解析：

```
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = etree.tostring(html)
print(result.decode('utf-8'))
```

test.html 的内容就是上面例子的 HTML 代码，内容如下：

```
<div>
<ul>
<li class="item-0"><a href="link1.html">first item</a></li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-inactive"><a href="link3.html">third item</a></li>
<li class="item-1"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a>
</ul>
</div>
```

这次输出结果略有不同，多了一个 DOCTYPE 声明，不过对解析没有任何影响，结果如下：

```
<html><body><div>
<ul>
<li class="item-0"><a href="link1.html">first item</a></li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-inactive"><a href="link3.html">third item</a></li>
<li class="item-1"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a>
</li></ul>
</div></body></html>
```

在python中使用xpath

```
from lxml import etree

# 第一种方式，直接在python代码中解析html字符串
text = """
<div>
  <ul>
    <li class="item-0"><a href="link1.html">first item</a></li>
    ....
  </ul>
</div>
"""
resp_html = etree.HTML(text)

#第二种方式，读取一个html文件并解析
html = etree.parse('./test.html', etree.HTMLParser())
result = etree.tostring(html)
print(result.decode('utf-8'))
```

5. 所有节点

用以 // 开头的 XPath 规则来选取所有符合要求的节点：

```
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//*')
print(result)

# 运行结果
"""
[<Element html at 0x1d6610ebe08>, <Element body at 0x1d6610ebf08>,
<Element div at 0x1d6610ebf48>, <Element ul at 0x1d6610ebf88>,
<Element li at 0x1d6610ebfc8>, <Element a at 0x1d661115088>,
<Element li at 0x1d6611150c8>, <Element a at 0x1d661115108>,
<Element li at 0x1d661115148>, <Element a at 0x1d661115048>,
<Element li at 0x1d661115188>, <Element a at 0x1d6611151c8>,
<Element li at 0x1d661115208>, <Element a at 0x1d661115248>]
"""
```

* 代表匹配所有节点，返回的结果是一个列表，每个元素都是一个 Element 类型，后跟节点名称。也可以指定匹配的节点名称：

```
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//li')
print(result)

# 运行结果
[<Element li at 0x23fb219af08>, <Element li at 0x23fb219af48>, <Element li at
0x23fb219af88>,
<Element li at 0x23fb219afc8>, <Element li at 0x23fb21c5048>]

<Element li at 0x23fb219af08>
```

取出其中某个对象时可以直接用索引。

6. 子节点

通过 / 或 // 即可查找元素的子节点或子孙节点。选择 li 节点的所有直接 a 子节点：

```
from lxml import etree

html = etree.parse('.test.html', etree.HTMLParser())
result = html.xpath('//li/a')
print(result)
```

此处的 / 用来获取直接子节点，如果要获取所有子孙节点，将 / 换成 // 即可。

7. 父节点

知道子节点，查询父节点可以用 .. 来实现：

```
# 获得 href 属性为 link4.html 的 a 节点的父节点的 class 属性

# 方法一
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//a[@href="link4.html"]/../@class')
print(result)

# 方法二
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//a[@href="link4.html"]/parent::*/@class')
print(result)

# 运行结果：['item-1']
```

8. 属性匹配

匹配时可以用@符号进行属性过滤：

```
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//li[@class="item-inactive"]')
print(result)

# 运行结果：[<Element li at 0x2089793a3c8>]
```

9. 文本获取

有两种方法：一是获取文本所在节点后直接获取文本，二是使用 //。

```
# 第一种
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//li[@class="item-0"]/a/text()')
print(result)

# 第二种
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//li[@class="item-0"]//text()')
print(result)
```

第二种方法会获取到补全代码时换行产生的特殊字符，推荐使用第一种方法，可以保证获取的结果是整洁的。

10. 属性获取

在 XPath 语法中，@符号相当于过滤器，可以直接获取节点的属性值：

```
from lxml import etree

html = etree.parse('./test.html', etree.HTMLParser())
result = html.xpath('//li/a/@href')
print(result)

# 运行结果：['link1.html', 'link2.html', 'link3.html', 'link4.html',
'link5.html']
```

11. 属性多值匹配

有时候，某些节点的某个属性可能有多个值：

```
from lxml import etree

text = '''
<li class="li li-first"><a href="link.html">first item</a></li>
'''
html = etree.HTML(text)
result = html.xpath('//li[contains(@class, "li")]/a/text()')
print(result)

# 运行结果：['first item']
```

12. 多属性匹配

当前节点有多个属性时，需要同时进行匹配：

```
from lxml import etree

text = '''
<li class="li li-first" name="item"><a href="link.html">first item</a></li>
'''
html = etree.HTML(text)
result = html.xpath('//li[contains(@class, "li") and @name="item"]/a/text()')
print(result)

# 运行结果：['first item']
```

扩展：XPath 运算符

运算符	描述	实例	返回值
or	或	age=18 or age=20	age=18：True；age=21：False
and	与	age>18 and age<21	age=20：True；age=21：False
mod	计算除法的余数	5 mod 2	1
|	计算两个节点集	//book | //cd	返回所有拥有 book 和 cd 元素的节点集
+	加法	5 + 3	8
-	减法	5 - 3	2
*	乘法	5 * 3	15
div	除法	8 div 4	2
=	等于	age=19	判断简单，不再赘述
!=	不等于	age!=19	判断简单，不再赘述
<	小于	age<19	判断简单，不再赘述
<=	小于等于	age<=19	判断简单，不再赘述
>	大于	age>19	判断简单，不再赘述
>=	大于等于	age>=19	判断简单，不再赘述

13. 按序选择

匹配结果有多个节点，需要选中第二个或最后一个，可以按照中括号内加索引或其他相应语法获得：

```
from lxml import etree
```

```
text = '''
<div>
<ul>
<li class="item-0"><a href="link1.html">first item</a></li>
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-inactive"><a href="link3.html">third item</a></li>
<li class="item-1"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a>
</ul>
</div>
'''

html = etree.HTML(text)
# 获取第一个
result = html.xpath('//li[1]/a/text()')
print(result)
# 获取最后一个
result = html.xpath('//li[last()]/a/text()')
print(result)
# 获取前两个
result = html.xpath('//li[position()<3]/a/text()')
print(result)
# 获取倒数第三个
result = html.xpath('//li[last()-2]/a/text()')
print(result)

"""
运行结果:
['first item']
['fifth item']
['first item', 'second item']
['third item']
"""
```

> XPath 中提供了100多个函数，包括存取、数值、逻辑、节点、序列等处理功能，具体作用可以参考： http://www.w3school.com.cn/xpath/xpath_functions.asp

14. 节点轴选择

XPath 提供了很多节点轴选择方法，包括子元素、兄弟元素、父元素、祖先元素等：

```
from lxml import etree

text = '''
<div>
<ul>
<li class="item-0"><a href="link1.html"><span>first item</span></a></li>
```

```
<li class="item-1"><a href="link2.html">second item</a></li>
<li class="item-inactive"><a href="link3.html">third item</a></li>
<li class="item-1"><a href="link4.html">fourth item</a></li>
<li class="item-0"><a href="link5.html">fifth item</a>
</ul>
</div>
'''

html = etree.HTML(text)
# 获取所有祖先节点
result = html.xpath('//li[1]/ancestor::*')
print(result)
# 获取 div 祖先节点
result = html.xpath('//li[1]/ancestor::div')
print(result)
# 获取当前节点所有属性值
result = html.xpath('//li[1]/attribute::*')
print(result)
# 获取 href 属性值为 link1.html 的直接子节点
result = html.xpath('//li[1]/child::a[@href="link1.html"]')
print(result)
# 获取所有的的子孙节点中包含 span 节点但不包含 a 节点
result = html.xpath('//li[1]/descendant::span')
print(result)
# 获取当前所有节点之后的第二个节点
result = html.xpath('//li[1]/following::*[2]')
print(result)
# 获取当前节点之后的所有同级节点
result = html.xpath('//li[1]/following-sibling::*')
print(result)

"""
[<Element html at 0x231a8965388>, <Element body at 0x231a8965308>, <Element
div at 0x231a89652c8>, <Element ul at 0x231a89653c8>]
[<Element div at 0x231a89652c8>]
['item-0']
[<Element a at 0x231a89653c8>]
[<Element span at 0x231a89652c8>]
[<Element a at 0x231a89653c8>]
[<Element li at 0x231a8965308>, <Element li at 0x231a8965408>, <Element li at
0x231a8965448>, <Element li at 0x231a8965488>]
"""
```

> 更多参考文档： 轴的用法：http://www.w3school.com.cn/xpath/xpath_axes.asp XPath 的用法：http://www.w3school.com.cn/xpath/index.asp Python lxml 的用法：http://lxml.de