4.2 scrapy 中查找 HTML 元素

在前面我们已经知道使用 BeautifulSoup 能查找 HTML 中的元素，scrapy 中也有强大的查找 HTML 元素的功能，那就是使用 xpath 方法。xpath 方法使用 XPath 语法，比 BeautifulSoup 的 select 要灵活而且速度快。

4.2.1 scrapy 的 xpath 简介

例 4-2-1: 使用 xpath 查找 HTML 中的元素

```
from scrapy.selector import Selector
htmlText='''
<html><body>
<bookstore>
<book>
   <title lang="eng">Harry Potter</title>
   <price>29.99</price>
</book>
<book>
   <title lang="eng">Learning XML</title>
   <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
print(type(selector));
print(selector)
s=selector.xpath("//title")
print(type(s))
print(s)
```

程序结果：

```
class 'scrapy.selector.unified.Selector'>
<Selector xpath=None data='<html><body>\n<bookstore>\n<book>\n   <title'>
<class 'scrapy.selector.unified.SelectorList'>
[<Selector xpath='//title' data='<title lang="eng">Harry Potter</title>'>, <Selector xpath='//title' data='<title lang="eng">Learning XML</title>'>]
```

我们来分析程序的功能：

（1） from scrapy.selector import Selector

从 scrapy 中引入 Selector 类，这个类就是选择查找类。

(2) selector=Selector(text=htmlText)

使用 htmlText 的文字建立 Selector 类，就是装载 HTML 文档，文档装载后就形成一个 Selector 对象，就可以使用 xpath 查找元素。

(3) print(type(selector)

可看到 selector 是一个类型为 scrapy.selector.unified.Selector，这个类型是一个有 xpath 方法的类型。

(4) s=selector.xpath("//title")

这个方法在文档中查找所有的<title>的元素，其中"//"表示文档中的任何位置。一般地：

selector.xpath("//tagName")

表示在权文档中搜索<tagName>的 tags，形成一个 Selector 的列表。

(5) print(type(s))

由于<title>有两个元素，因此我们看到这是一个 scrapy.selector.unified.SelectorList 类，类似 scrapy.selector.unified.Selector 的列表。

(6) print(s)

我们看到 s 包含两个 Selector 对象，一个是<Selector xpath='//title' data='<title lang="eng">Harry Potter</title>'>， 另外一个是<Selector xpath='//title' data='<title lang="eng">Learning XML</title>'>。

<Selector xpath='//title' data='<title lang="eng">Learning XML</title>'>

由此可见一般 selector 搜索一个<tagName>的 HTML 元素的方法是：

selector.xpath("//tagName")

在装载 HTML 文档后 selector=Selector(text=htmlText)得到的 selector 是对应全文档顶层的元素<html>的，其中"//"表示全文档搜索，结果是一个 Selector 的列表，哪怕只有一个元素也成一个列表，例如：

selector.xpath("//body") 搜索到<body>元素，结果是一个 Selector 的列表，包含一个 Selector 元素；

selector.xpath("//title")搜索到两个<title>元素，结果是 Selector 的列表，包含 2 个 Selector 元素；

selector.xpath("//book")搜索到两个<book>元素，结果是 Selector 的列表，包含 2 个 Selector 元素；

4.2.2 xpath 查找 HTML 元素

1、使用"//"表示文档下面的所有节点元素，用"/"表示当前节点的下一级节点元素。

例 4-2-2："//"与"/"的使用

selector.xpath("//bookstore/book") 搜索<bookstore>下一级的<book>元素，找到 2 个；

selector.xpath("//body/book") 搜索<body>下一级的<book>元素，结果为空；

selector.xpath("//body//book") 搜索<body>下<book>元素，找到 2 个；

selector.xpath("/body//book") 搜索文档下一级的<body>下的<book>元素，找结果为空，因为文档的下一级是<html>元素，不是<body>元素；

selector.xpath("/html/body//book")或者 selector.xpath("/html//book") 搜索<book>元素，找到 2 个；

selector.xpah("//book/title") 搜索文档中所有<book>下一级的<title>元素，找到 2 个，结果与 selector.xpah("//title")、selector.xpath("//bookstore//title")一样；

selector.xpath("//book//price")与 selector.xpath("//price")结果一样，都是找到 2 个<price>元素；

2、使用"."表示当前节点元素，使用 xpath 可以连续调用，如果前一个 xpath 返回一个

Selector 的列表，那么这个列表可以继续调用 xpath，功能是为每个列表元素调用 xpath，最后结果是全部元素调用 xpath 的汇总。

例 4-2-3：使用"."进行 xpath 连续调用

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
<bookstore>
<title>books</title>
<book>
  <title>Novel</title>
  <title lang="eng">Harry Potter</title>
  <price>29.99</price>
</book>
<book>
  <title>TextBook</title>
  <title lang="eng">Learning XML</title>
  <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book").xpath("./title")
for e in s:
    print(e)
```

程序结果：

```
<Selector xpath='//book/title' data='<title>Novel</title>'>
<Selector xpath='//book/title' data='<title lang="eng">Harry Potter</title>'>
<Selector xpath='//book/title' data='<title>TextBook</title>'>
<Selector xpath='//book/title' data='<title lang="eng">Learning XML</title>'>
```

我们看到 selector.xpath("//book")首先搜索到文档中所有<book>元素，总共有 2 个，然后再次调用 xpath("./title")，就是从当前元素<book>开始往下一级搜索<title>，每个<book>都找到 2 个<title>，因此结果有 4 个<title>。

注意如果 xpath 连续调用时不指定是从前一个 xpath 的结果元素开始的，那么默认是从全文档开始的，结果会不一样，例如：

```
s=selector.xpath("//book").xpath("/title")
```

结果是空的，因为后面的 xpath("/title")从文档开始搜索<title>。

```
s=selector.xpath("//book").xpath("//title")
```

结果有 10 个元素，因为每个<book>都驱动 xpath("//title")在全文档搜索<title>元素，每次都搜索到 5 个元素。

3、如果 xpath 返回的 Selector 对象列表，再次调用 extract()函数会得到这些对象的元素文本的列表，使用 extract_first()获取列表中第一个元素值，如果列表为空 extract_first()的值为 None。

而对于单一的一个 Selector 对象，调用 extract()函数就可以得到 Selector 对象对应的元素的文本值。单一的 Selector 对象没有 extract_first()函数。

例 4-2-4：extract 与 extract_first 函数使用

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
<bookstore>
<book id="b1">
   <title lang="english">Harry Potter</title>
   <price>29.99</price>
</book>
<book id="b2">
   <title lang="chinese">学习 XML</title>
   <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book/price")
print(type(s),s)
s=selector.xpath("//book/price").extract()
print(type(s),s)
s=selector.xpath("//book/price").extract_first()
print(type(s),s)
```

程序结果：

<class 'scrapy.selector.unified.SelectorList'> [<Selector xpath='//book/price' data='<price>29.99</price>'>, <Selector xpath='//book/price' data='<price>39.95</price>'>]

<class 'list'> ['<price>29.99</price>', '<price>39.95</price>']

<class 'str'> <price>29.99</price>

由此可见:

s=selector.xpath("//book/price") 得到的是 SelectorList 列表；

s=selector.xpath("//book/price").extract() 得到的是<price>元素的 Selector 对象对应的<price>元素的文本组成的列表，即：

['<price>29.99</price>', '<price>39.95</price>']

s=selector.xpath("//book/price").extrac_first() 得到的是<price>元素的文本组成的列表的第一个元素，是一个文本，即：

<price>29.99</price>

4、xpath 使用"/@attrName"得到一个 Selector 元素的 attrName 属性节点对象，属性节点对象也是一个 Selector 对象，通过 extract()获取属性值。

例 4-2-5：获取元素属性值

```
htmlText='''
<html>
<body>
<bookstore>
<book id="b1">
   <title lang="english">Harry Potter</title>
   <price>29.99</price>
</book>
<book id="b2">
   <title lang="chinese">学习 XML</title>
   <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book/@id")
print(s)
print(s.extract())
for e in s:
    print(e.extract())
```

程序结果：

```
[<Selector xpath='//book/@id' data='b1'>, <Selector xpath='//book/@id' data='b2'>]
['b1', 'b2']
b1
b2
```

由此可见：

s=selector.xpath("//book/@id")

结果是 2 个<book>的 id 属性组成的 SelectorList 列表，即属性也是一个 Selector 对象；

print(s.extract())

结果是<book>的 id 属性的两个 Selector 对象的属性文本值的列表，即['b1', 'b2']；

```
for e in s:
    print(e.extract())
```

每个 e 是一个 Selector 对象，因此 extract()获取对象的属性值。

5、xpath 使用"/text()"得到一个 Selector 元素包含的文本值，文本值节点对象也是一个 Selector 对象，通过 extract()获取文本值。

例：4-2-6：获取节点的文本值

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
<bookstore>
<book id="b1">
   <title lang="english">Harry Potter</title>
   <price>29.99</price>
</book>
<book id="b2">
   <title lang="chinese">学习 XML</title>
   <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book/title/text()")
print(s)
print(s.extract())
for e in s:
    print(e.extract())
```

程序结果：

[<Selector xpath='//book/title/text()' data='Harry Potter'>, <Selector xpath='//book/title/text()' data='学习 XML'>]

['Harry Potter', '学习 XML']

Harry Potter

学习 XML

由此可见：

s=selector.xpath("//book/title/text()")

结果也是 SelectorList 列表，即文本也是一个节点；

print(s.extract())

结果是文本节点的字符串值的列表，即['Harry Potter', '学习 XML']；

```
for e in s:
    print(e.extract())
```

每个 e 是一个 Selector 对象，因此 extract()获取对象的属性值。

值得注意的是如果一个 element 的元素包含的文本不是单一的文本，那么可能会产生多个文本值。

例 4-2-7： 多个文本节点值

```
from scrapy.selector import Selector
```

```
htmlText='''
<html>
<body>
<bookstore>
<book id="b1">
  <title lang="english"><b>H</b>arry <b>P</b>otter</title>
  <price>29.99</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book/title/text()")
print(s)
print(s.extract())
for e in s:
    print(e.extract())
```

程序结果：

[<Selector xpath='//book/title/text()' data='arry '>, <Selector xpath='//book/title/text()' data='otter'>]

['arry ', 'otter']

arry

otter

由此可见<title<中的文本值包含 arry 与 otter 两个。

6、xpath 使用"tag[condition]"来限定一个 tag 元素，其中 condition 是由这个 tag 的属性、文本等计算出的一个逻辑值。如果有多个条件，那么可以写成：

"tag[condition1][condition2]...[conditionN]"

或者：

"tag[condition1 and condition2 and ... and conditionN]"

例 4-2-8：使用 condition 限定 tag 元素

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
<bookstore>
<book id="b1">
  <title lang="english">Harry Potter</title>
  <price>29.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
```

```
    <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book/title[@lang='chinese']/text()")
print(s.extract_first())
s=selector.xpath("//book[@id='b1']/title")
print(s.extract_first())
```

程序结果：
学习 XML
<title lang="english">Harry Potter</title>
由此可见：
s=selector.xpath("//book/title[@lang='chinese']/text()")
搜索<book>下面属性 lang="chinese"的<title>
s=selector.xpath("//book[@id='b1']/title")
搜索属性 id="b1"的<book>下面的<title>。

7、xpath 可以使用 position()来确定其中一个元素的限制，这个选择序号是从 1 开始的，不是从 0 开始编号的，还可以通过 and、or 等构造复杂的表达式。

例 4-2-9：使用 position()序号来确定锁选择的元素

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
<bookstore>
<book id="b1">
  <title lang="english">Harry Potter</title>
  <price>29.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
  <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book[position()=1]/title")
print(s.extract_first())
s=selector.xpath("//book[position()=2]/title")
```

```
print(s.extract_first())
```

程序结果：

```
<title lang="english">Harry Potter</title>
<title lang="chinese">学习 XML</title>
```

其中：

```
s=selector.xpath("//book[position()=1]/title")
s=selector.xpath("//book[position()=1]/title")
```

分别选择第一、二个<book>元素。

8、xpath 使用星号"*"代表任何 Element 节点，不包括 Text、Comment 的节点。

例 4-2-10： 使用"*"代表任何 element 元素

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
<bookstore>
<book id="b1">
  <title lang="english">Harry Potter</title>
  <price>29.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
  <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//bookstore/*/title")
print(s.extract())
```

程序结果：

['<title lang="english">Harry Potter</title>', '<title lang="chinese">学习 XML</title>']

其中 s=selector.xpath("//bookstore/*/title")是搜索<bookstore>的孙子节点<title>，中间隔开一层任何元素。

9、xpath 使用"@*"代表任何属性

例 4-2-11：使用@*代表属性

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
```

```
<bookstore>
<book>
  <title lang="english">Harry Potter</title>
  <price>29.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
  <price>39.95</price>
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//book[@*]/title")
print(s.extract())
s=selector.xpath("//@*")
print(s.extract())
```

程序结果：

```
['<title lang="chinese">学习 XML</title>']
['english', 'b2', 'chinese']
```

其中：

s=selector.xpath("//book[@*]/title")

是搜索任何包含属性的<book>元素下面的<title>，结果搜索到第二个<book>

s=selector.xpath("//@*")

是搜索文档中所有属性节点。

10、xpath 使用"element/parent::*"选择 element 的父节点，这个节点只有一个。如果写成 element/parent::tag，就指定 element 的 tag 父节点，除非 element 的父节点正好为><tag>节点，不然就为 None。

例 4-2-12： xpath 搜索元素的父节点

```
from scrapy.selector import Selector
htmlText='''
<html>
<body>
<bookstore>
<book>
  <title lang="english">Harry Potter</title>
  <price>29.99</price>
</book>
<book id="b2">
  <title lang="chinese">学习 XML</title>
  <price>39.95</price>
```

```
</book>
</bookstore>
</body></html>
'''
selector=Selector(text=htmlText)
s=selector.xpath("//title[@lang='chinese']/parent::*")
print(s.extract())
```

程序结果：

['<book id="b2">\n <title lang="chinese">学习 XML</title>\n <price>39.95</price>\n</book>']

其中 s=selector.xpath("//title[@lang='chinese']/parent::*")是查找属性为 lang='chinese'的<title>元素的父节点，就是 id="b2"的<book>元素节点。

11、xpath 使用"element/folllowing-sibling::*"搜索 element 后面的同级的所有兄弟节点，使用"element/folllowing-sibling::*[position()=1]" 搜索 element 后面的同级的第一个兄弟节点。

例 4-2-13： 搜索后面的兄弟节点

```
from scrapy.selector import Selector
htmlText="<a>A1</a><b>B1</b><c>C1</c><d>D<e>E</e></d><b>B2</b><c>C2</c>"
selector=Selector(text=htmlText)
s=selector.xpath("//a/following-sibling::*")
print(s.extract())
s=selector.xpath("//a/following-sibling::*[position()=1]")
print(s.extract())
s=selector.xpath("//b[position()=1]/following-sibling::*")
print(s.extract())
s=selector.xpath("//b[position()=1]/following-sibling::*[position()=1]")
print(s.extract())
```

程序结果：

['B1', '<c>C1</c>', '<d>D<e>E</e></d>', 'B2', '<c>C2</c>']
['B1']
['<c>C1</c>', '<d>D<e>E</e></d>', 'B2', '<c>C2</c>']
['<c>C1</c>']

例如：

s=selector.xpath("//b[position()=1]/following-sibling::*[position()=1]")

是搜索第一个节点后面的第一个兄弟节点，即<c>C1</c>节点。

12、xpath 使用"element/preceding-sibling::*"搜索 element 前面的同级的所有兄弟节点，使用"element/preceding-sibling::*[position()=1]" 搜索 element 前面的同级的第一个兄弟节点。

例 4-2-14： 搜索前面的兄弟节点

```
from scrapy.selector import Selector
htmlText="<a>A1</a><b>B1</b><c>C1</c><d>D<e>E</e></d><b>B2</b><c>C2</c>"
selector=Selector(text=htmlText)
s=selector.xpath("//a/preceding-sibling::*")
print(s.extract())
s=selector.xpath("//b/preceding-sibling::*[position()=1]")
print(s.extract())
s=selector.xpath("//b[position()=2]/preceding-sibling::*")
print(s.extract())
s=selector.xpath("//b[position()=2]/preceding-sibling::*[position()=1]")
print(s.extract())
```

程序结果：

```
[]
['<a>A1</a>', '<d>D<e>E</e></d>']
['<a>A1</a>', '<b>B1</b>', '<c>C1</c>', '<d>D<e>E</e></d>']
['<d>D<e>E</e></d>']
```

例如：

```
s=selector.xpath("//b/preceding-sibling::*[position()=1]")
```

是所有<b>前面的第一个兄弟节点，因为有 2 个<b>节点，因此结果是['<a>A1', '<d>D<e>E</e></d>']

XPath 在 HTML 文档中搜索元素是一个技术标志，限于篇幅这里只是讲解了主要的规则，还有一些规则可以查询得到，关键是要在实践中慢慢熟悉。