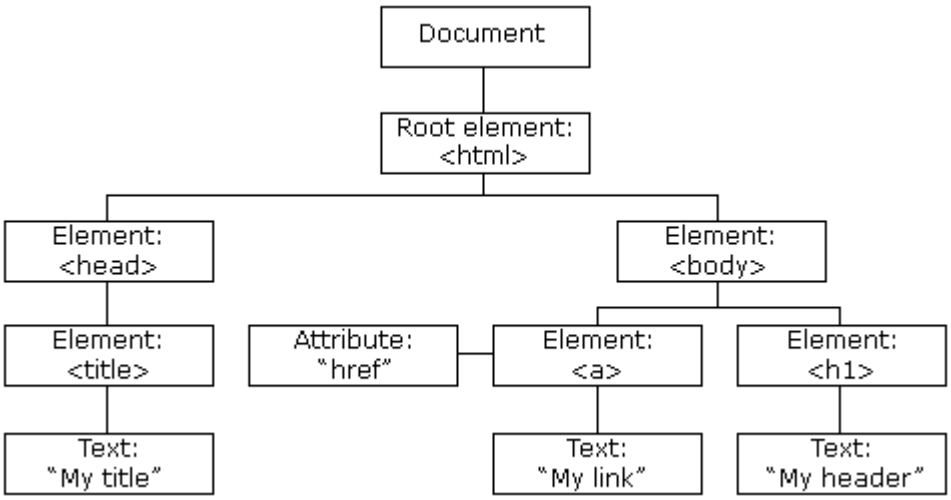
# Xpath

## HTML DOM 模型示例

HTML DOM 定义了访问和操作 HTML 文档的标准方法，以树型结构表示 HTML 文档。




## 如何实现爬虫数据的解析?

1、定位 html 文档中的节点

2、提取指定节点的属性，比如 href,class 等

3、获取指定节点的文本，比如 a、p，div,span，div 等的文本

# 什么是 XPath?

XPath (XML Path Language) 是一门在 XML 文档中查找信息的语言，可用来在 XML 文档中对元素和属性进行遍历。

XPath 可以先将 HTML 文件 转换成 XML 文档,然后用 XPath 查找 HTML 节点或元素。

# 什么是 XML

- 可扩展标记语言（EXtensible Markup Language），W3C 的推荐标准，类似 HTML，标签需要我们自行定义，具有自我描述性，设计宗旨是传输数据，而非显示数据

## XML 和 HTML 的区别

| 数据格式 | 描述 | 设计目标 |
| --- | --- | --- |
| XML | Extensible Markup Language （可扩展标记语言） | 被设计为传输和存储数据，其焦点是数据的内容。 |
| HTML | HyperText Markup Language （超文本标记语言） | 如何更好显示数据。 |
| HTML DOM | Document Object Model for HTML (文档对象模型) | 通过 HTML DOM，可以访问所有的 HTML 元素，连同它们所包含的文本和属性。可以对其中的内容进行修改和删除，同时也可以创建新的元素。 |

# XPath 语法

## 选取节点

XPath 使用路径表达式来选取 XML 文档中的节点或者节点集。这些路径表达式和我们在常规的电脑文件系统中看到的表达式非常相似。

下面列出了最常用的路径表达式：

| 表达式 | 描述 |
| --- | --- |
| nodename | 选取此节点的所有子节点。 |
| / | 从根节点选取。 |
| // | 从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。 |
| . | 选取当前节点。 |
| .. | 选取当前节点的父节点。 |
| @ | 选取属性。 |

一些路径表达式以及表达式的结果：

| 路径表达式 | 结果 |
| --- | --- |
| bookstore | 选取 bookstore 元素的所有子节点。 |
| /bookstore | 选取根元素 bookstore。注释：假如路径起始于正斜杠( / )，则此路径始终代表到某元素的绝对路径！ |
| bookstore/book | 选取属于 bookstore 的子元素的所有 book 元素。 |
| //book | 选取所有 book 子元素，而不管它们在文档中的位置。 |
| bookstore//book | 选择属于 bookstore 元素的后代的所有 book 元素，而不管它们位于 bookstore 之下的什么位置。 |
| //@lang | 选取名为 lang 的所有属性。 |

# 谓语（Predicates）

谓语用来查找某个特定的节点或者包含某个指定的值的节点，被嵌在方括号中。

在下面的表格中，我们列出了带有谓语的一些路径表达式，以及表达式的结果：

| 路径表达式 | 结果 |
| --- | --- |
| /bookstore/book[1] | 选取属于 bookstore 子元素的第一个 book 元素。 |
| /bookstore/book[last()] | 选取属于 bookstore 子元素的最后一个 book 元素。 |
| /bookstore/book[last()-1] | 选取属于 bookstore 子元素的倒数第二个 book 元素。 |
| /bookstore/book[position()<3] | 选取最前面的两个属于 bookstore 元素的子元素的 book 元素。 |
| //title[@lang] | 选取所有拥有名为 lang 的属性的 title 元素。 |
| //title[@lang='eng'] | 选取所有 title 元素，且这些元素拥有值为 eng 的 lang 属性。 |
| /bookstore/book[price>35.00] | 选取 bookstore 元素的所有 book 元素，且其中的 price 元素的值须大于 35.00。 |
| /bookstore/book[price>35.00]/title | 选取 bookstore 元素中的 book 元素的所有 title 元素，且其中的 price 元素的值须大于 35.00。 |
| // title [contains(@lang,"e")] | 选取所有 title 元素，且 lang 属性包含 e |

# 选取未知节点

XPath 通配符可用来选取未知的 XML 元素。

| 通配符 | 描述 |
| --- | --- |
| * | 匹配任何元素节点。 |

| 通配符 | 描述 |
| --- | --- |
| @* | 匹配任何属性节点。 |
| node() | 匹配任何类型的节点。 |

在下面的表格中列出了一些路径表达式，以及这些表达式的结果：

| 路径表达式 | 结果 |
| --- | --- |
| /bookstore/* | 选取 bookstore 元素的所有子元素。 |
| //* | 选取文档中的所有元素。 |
| //title[@*] | 选取所有带有属性的 title 元素。 |

# 选取若干路径

通过在路径表达式中使用“|”运算符，可以选取若干个路径。

| 路径表达式 | 结果 |
| --- | --- |
| //book/title \| //book/price | 选取 book 元素的所有 title 和 price 元素。 |
| //title \| //price | 选取文档中的所有 title 和 price 元素。 |
| /bookstore/book/title \| //price | 选取属于 bookstore 元素的 book 元素的所有 title 元素，以及文档中所有的 price 元素。 |

# XPath 的运算符

下面列出了可用在 XPath 表达式中的运算符：

| 运算符 | 描述 | 实例 | 返回值 |
| --- | --- | --- | --- |
| \| | 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 | 6 + 4 | 10 |
| - | 减法 | 6 - 4 | 2 |
| * | 乘法 | 6 * 4 | 24 |
| div | 除法 | 8 div 4 | 2 |
| = | 等于 | price=9.80 | 如果 price 是 9.80，则返回 true。如果 price 是 9.90，则返回 false。 |
| != | 不等于 | price!=9.80 | 如果 price 是 9.90，则返回 true。如果 price 是 9.80，则返回 false。 |
| < | 小于 | price<9.80 | 如果 price 是 9.00，则返回 true。如果 price 是 9.90，则返回 false。 |
| <= | 小于或等于 | price<=9.80 | 如果 price 是 9.00，则返回 true。如果 price 是 9.90，则返回 false。 |
| > | 大于 | price>9.80 | 如果 price 是 9.90，则返回 true。如果 price 是 9.80，则返回 false。 |
| >= | 大于或等于 | price>=9.80 | 如果 price 是 9.90，则返回 true。如果 price 是 9.70，则返回 false。 |
| or | 或 | price=9.80 or price=9.70 | 如果 price 是 9.80，则返回 true。如果 price 是 9.50，则返回 false。 |
| and | 与 | price>9.00 and price<9.90 | 如果 price 是 9.80，则返回 true。如果 price 是 8.50，则返回 false。 |
| mod | 计算除法的余数 | 5 mod 2 | 1 |

# lxml 库

lxml 是一款高性能的 Python HTML/XML 的解析器，用 C 实现的，主要的功能是利用 XPath 语法解析和提取 HTML/XML 数据，来快速的定位特定元素以及节点信息。

lxml python 官方文档：http://lxml.de/index.html

# 安装

pip install lxml

# 解析 HTML 代码

```
# 使用 lxml 的 etree 库
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> # 注意，此处缺少一个 </li> 闭合标签
     </ul>
 </div>
'''

#利用 etree.HTML，将字符串解析为 HTML 文档
html = etree.HTML(text)

# 按字符串序列化 HTML 文档
result = etree.tostring(html).decode()

print(result)
```

lxml 可以自动修正 html 代码，例子里不仅补全了 li 标签，还添加了 body，html 标签。

# html 文件解析

hello.html 文件：

```
<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"><span
class="bold">third item</span></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>

# lxml_parse.py

from lxml import etree

# 读取外部文件 hello.html
html = etree.parse('./hello.html')
result = etree.tostring(html, pretty_print=True).decode()
print(result)
```

# XPath 实例测试

## 1. 获取所有的<li>标签

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')
print(type(html))  # 显示 etree.parse() 返回类型

result = html.xpath('//li')

print(result)  # 打印<li>标签的元素集合
print(len(result))
print(type(result))
print(type(result[0]))
```

## 2. 获取<li> 标签下的所有  标签

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')
```

```
#result = html.xpath('//li/span')
#注意这么写是不对的：
#因为 / 是用来获取子元素的，而 <span> 并不是 <li> 的子元素，所以，要用双斜杠

result = html.xpath('//li//span')

print(result)
```

## 3. 获取<li>标签下 href 为 link1.html 的 <a> 标签

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')
result = html.xpath('//li/a[@href="link1.html"]')

print(result)
```

## 4. 获取<li> 标签的所有 class 属性

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')
result = html.xpath('//li/@class')

print(result)
```

**运行结果**

```
[<Element a at 0x10ffaae18>]
```

## 5. 获取 <li> 标签下的<a>标签里的所有 class 属性

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')
result = html.xpath('//li/a//@class')

print(result)
```

## 6. 获取最后一个 <li> 的 <a> 的 href

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')

result = html.xpath('//li[last()]/a/@href')
# 谓语 [last()] 可以找到最后一个元素

print(result)
```

## 7. 获取倒数第二个元素的内容

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')
result = html.xpath('//li[last()-1]/a')

# text 方法可以获取元素内容
print(result[0].text)
```

## 8. 获取 class 值为 bold 的标签名

```
# xpath_li.py

from lxml import etree

html = etree.parse('hello.html')

result = html.xpath('//*[@class="bold"]')

# tag 方法可以获取标签名
print(result[0].tag)
```

## 思考题

### 1、Python 中单引号，双引号，3 个单引号及 3 个双引号的区别

当你用单引号' '定义字符串的时候，它就会认为你字符串里面的双引号" "是普通字符，从而

不需要转义。反之当你用双引号定义字符串的时候，就会认为你字符串里面的单引号是普通字符无需转义。3 个引号实现多行输出效果或者加注释

**2、xpath 中/和//的区别**

/用来获取子元素，//用来获取子孙后代的元素

**3、如何获取节点内容?**

用 text 选取文本内容。

4、**如何获取节点属性?**

用@选取属性。