xpath与json

有同学说，我正则用的不好，处理HTML文档很累，有没有其他的方法?

有！那就是XPath，我们可以先将 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

- XML 指可扩展标记语言（EXtensible Markup Language）
- XML 是一种标记语言，很类似 HTML
- XML 的设计宗旨是传输数据，而非显示数据
- XML 的标签需要我们自行定义。
- XML 被设计为具有自我描述性。
- XML 是 W3C 的推荐标准

W3School官方文档：http://www.w3school.com.cn/xml/index.asp

XML 和 HTML 的区别

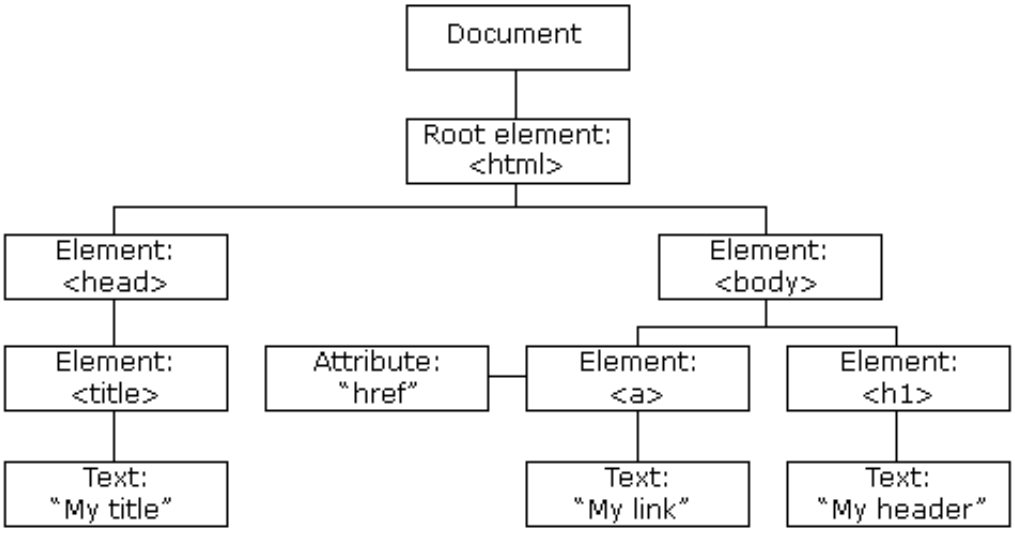
数据格式	描述	设计目标
XML	Extensible Markup Language（可扩展标记语言）	被设计为传输和存储数据，其焦点是数据的内容。
HTML	HyperText Markup Language（超文本标记语言）	显示数据以及如何更好显示数据。
HTML DOM	Document Object Model for HTML (文档对象模型)	通过 HTML DOM，可以访问所有的 HTML 元素，连同它们所包含的文本和属性。可以对其中的内容进行修改和删除，同时也可以创建新的元素。

```xml
<?xml version="1.0" encoding="utf-8"?>

<bookstore>

  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
```

```
    <price>29.99</price>
  </book>

  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price>
  </book>

  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>

</bookstore>
```

HTML DOM 模型示例

HTML DOM 定义了访问和操作 HTML 文档的标准方法，以树结构方式表达 HTML 文档。



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。

W3School官方文档：http://www.w3school.com.cn/xpath/index.asp

Xpath常用语法

表达式	说明	示例
/	从根节点开始选取	/html/body/div[3]/div/div[1]/div[2]/div[1]/div/h2
//	选择所有指定的结点	//div
.	当前结点	./div
..	父节点	../div
@	选取属性	//a/@href
text()	获取标签内的字符	//li/text()
node/elem[i]	选取node结点下的第i个elem元素 注意：下标从1开始。	ul/li[2]
node/elem[last()]	选取node结点下的最后一个elem元素	ul/li[last()]
//elem[@attr='xxx']	选取所有属性attr为xxx的elem元素	//div[@class='content']

初步使用

我们利用它来解析 HTML 代码，简单示例：

```
# 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)

print(result)
```

输出结果：

```
<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>
```

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。

文件读取：

除了直接读取字符串，lxml还支持从文件里读取内容。我们新建一个hello.html文件：

```
<!-- hello.html -->

<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>
```

再利用 etree.parse() 方法来读取文件。

```
from lxml import etree

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

print(result)
```

输出结果与之前相同：

```
<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>
```

XPath实例测试

<1> 获取所有的`<li>`标签

```
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> 继续获取`<li>`标签的所有 class属性

```
from lxml import etree

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

print(result)
```

运行结果

```
['item-0', 'item-1', 'item-inactive', 'item-1', 'item-0']
```

<3> 获取`<li>`标签下的所有`<span>`标签

```
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)
```

运行结果

```
[<Element span at 0x10d698e18>]
```

<4> 获取最后一个`<li>`的`<a>`的href

```
from lxml import etree

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

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

print(result)
```

运行结果

```
['link5.html']
```

<5> 获取倒数第二个元素的内容

```
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
```

运行结果

```
fourth item
```

JSON格式数据与json模块

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，它使得人们很容易的进行阅读和编写。同时也方便了机器进行解析和生成。适用于进行数据交互的场景，比如网站前台与后台之间的数据交互。

JSON和XML的比较可谓不相上下。

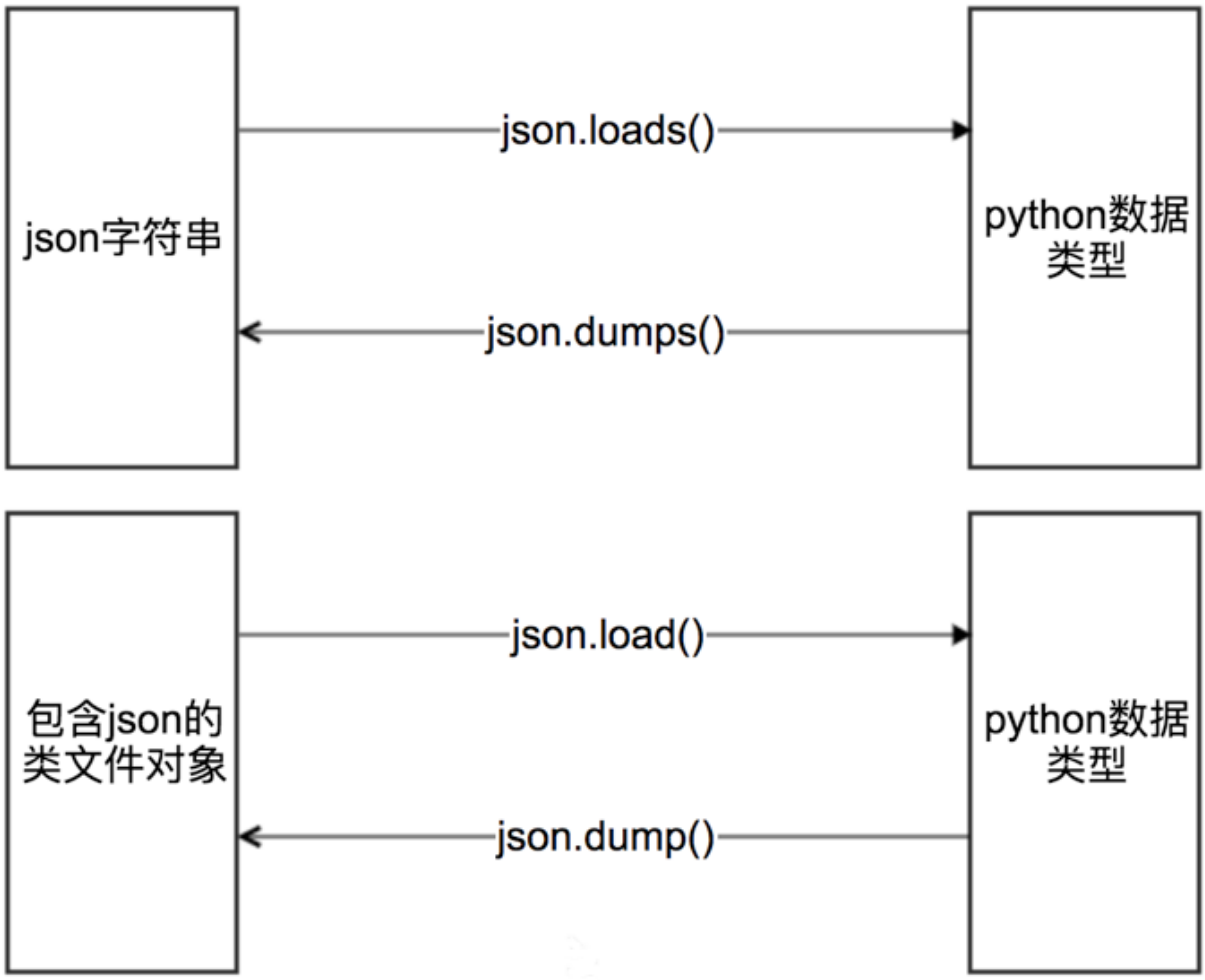
JSON

json简单说就是javascript中的对象和数组，所以这两种结构就是对象和数组两种结构，通过这两种结构可以表示各种复杂的结构

> 对象：对象在js中表示为{ }括起来的内容，数据结构为 { key：value, key：value, ... }的键值对的结构，在面向对象的语言中，key为对象的属性，value为对应的属性值，所以很容易理解，取值方法为 对象.key 获取属性值，这个属性值的类型可以是数字、字符串、数组、对象这几种。
>
> 数组：数组在js中是中括号[ ]括起来的内容，数据结构为 ["Python", "javascript", "C++", ...]，取值方式和所有语言中一样，使用索引获取，字段值的类型可以是 数字、字符串、数组、对象几种。

Python中的json模块

json模块提供了四个功能：dumps、dump、loads、load，用于字符串 和 python数据类型间进行转换。



简记: 只要和str类型有关的就加s，由str->python对象就是导入，用loads，由python对象非->str就是导出，用dumps

<1> json.loads()

把Json格式字符串解码转换成Python对象 从json到python的类型转化对照如下：

```
import json

str_li = "[1, 2, 3, 4]"

str_dict = '{"name": "JunGe", "city": "Changsha"}'

json.loads(str_li)

json.loads(str_dict)
```

<2> json.dumps()

实现python类型转化为json字符串，返回一个str对象 把一个Python对象编码转换成Json字符串

```
import json

list_obj = [1, 2, 3, 4]

dict_obj = {"name": "JunGe", "city": "Changsha"}

json.dumps(list_obj)

json.dumps(dict_obj)
```

<3> json.dump()

将Python内置类型序列化为json对象后写入文件

```
import json

list_obj = [{"city": "Changsha"}, {"name": "JunGe"}]
json.dump(list_obj, open("list_obj.json","w"),
ensure_ascii=False)

dict_obj = {"city": "Changsha", "name": "JunGe"}
json.dump(dict_obj, open("dict_obj.json","w"),
ensure_ascii=False)
```

<4> json.load()

读取文件中json形式的字符串元素 转化成python类型

```python
import json

list_obj = json.load(open("list_obj.json"))
print(list_obj)

dict_obj = json.load(open("dict_obj.json"))
print(dict_obj)
```