专栏：005：Beautiful Soup 的使用

E – 爬虫技术)(爬虫专栏

> 系列爬虫专栏
>
> 崇尚的学习思维是：输入，输出平衡，且平衡点不断攀升。
>
> 曾经有大神告诫说：没事别瞎写文章；所以，很认真的写的是能力范围内的，看客要是看不懂，不是你的问题，问题在我，得持续输入，再输出。
>
> 今天的主题是：BeautifulSoup解析文本

1：框架

序号	内容	说明
01	概念	-
02	函数方法	-
03	代码示例	-
04	博文实战	-
05	总结说明	—

2：概念

- 什么是BeautifulSoup?

BeautifulSoup 是一个可以从HTML或XML文件中提取数据的第三方python库。

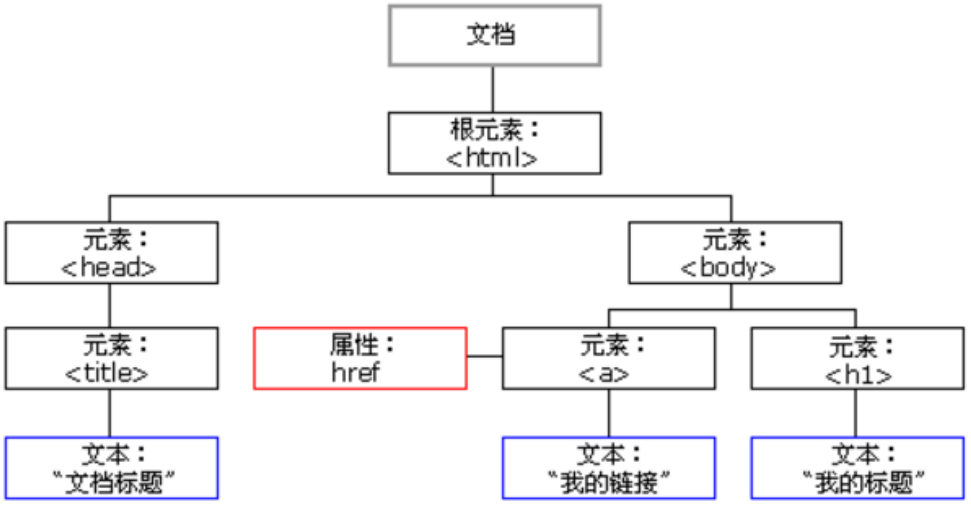
复述：是一个第三方库，所以需要自己安装。能从文本中解析所需要的文本。实现的功能和正则表达式一样，只不过方法不同。

- 什么是XML?

XML是指可扩展标记语言。被设计用来传输和存储数据。（这些和网页的知识有关，不懂，那算了）

- DOM 树？
  DOM是文档对象化模型（Document Object Model）的简称。DOM Tree是指通过DOM将HTML页面进行解析，并生成的HTML tree树状结构和对应访问方法。

一张图展现常见网页中出现的符号显示



解析文本常见的概念：

序号	概念	说明
01	Tag	标签
02	Name	名字
03	Attributes	属性

会涉及什么兄弟节点，父节点等概念。(不懂没关系，看看文档就知道什么意思)

3：代码示例

```
BeautifulSoup使用方法
BeautifulSoup(markup,"lxml",from_encoding ="utf-8")
```

第一个参数是需要解析的文本。
第二个参数是解析器的选择。lxml，所以需要安装第三方lxml库。
第三个参数是编码。中文，你懂的。

```
# -*- coding:utf-8 -*-
# To: learn BeautifulSoup
# Date: 2016.04.29
# Author: wuxiaoshen

from bs4 import BeautifulSoup

html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters;
and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie
</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie
</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Till
ie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""
Soup = BeautifulSoup(html_doc,'lxml',from_encoding='utf-8')
# 规格化输出：带缩进的输出
print(Soup.prettify())
```

```
# 还是上面的文本
Soup = BeautifulSoup(html_doc,  'lxml', from_encoding='utf-8')
# 获取标签、标签名字，标签内容
print(Soup.title)
# 输出：<title>The Dormouse's story</title>
print(Soup.title.name)
# 输出：title
print(Soup.title.string)
# 输出：The Dormouse's story

# 获取属性
print(Soup.p["class"])
# 输出：['title']

# 获取特定的全部标签

print(Soup.find_all('a')) # 返回一个list

# 输出：[<a class="sister" href="http://example.com/elsie" id="link
1">Elsie</a>, <a class="sister" href="http://example.com/lacie" id
="link2">Lacie</a>, <a class="sister" href="http://example.com/til
lie" id="link3">Tillie</a>]

print(Soup.find(id="link2") )
# 输出：<a class="sister" href="http://example.com/lacie" id="link
2">Lacie</a>

# 获取文档中所有的文字内容 方法：get_text()
print(Soup.get_text())
# 输出
---
The Dormouse's story

The Dormouse's story
Once upon a time there were three little sisters; and their names
were
Elsie,
Lacie and
Tillie;
and they lived at the bottom of a well.
...

---
# 大概看出了，是如何解析文本的了，如何获取标签，便签名字，属性等操作
```

大概的思路是：先下载网页源代码，得到一个BeautifulSoup对象。然后通过这些节点，便签，文本等获取你想要的信息。

经常使用的方法总结：

序号	方法	解释说明
01	find_all()	搜索全部符合要求的信息
02	get_text()	获取文本
03	find()	注意和find_all（ ）的区别

```
find( name , attrs , recursive , text , **kwargs )
find_all( name , attrs , recursive , text , **kwargs )
```

```
# 还是上面的文本信息
print(Soup.find('a')) # 返回一个list
print(Soup.a)
print(Soup.find_all('a'))

# output
<a class="sister" href="http://example.com/elsie" id="link1">Elsie
</a>
<a class="sister" href="http://example.com/elsie" id="link1">Elsie
</a>
[<a class="sister" href="http://example.com/elsie" id="link1">Elsi
e</a>, <a class="sister" href="http://example.com/lacie" id="link
2">Lacie</a>, <a class="sister" href="http://example.com/tillie" i
d="link3">Tillie</a>]

----
默认存在多个相同的节点属性不同，比如“a” ，默认查找第一个节点
```

更多信息查看文档

4：博文抓取实战

抓取任务：抓取一篇博客的全部文字信息，并保存至本地文本中。

```
url = http://blog.csdn.net/pongba/article/details/4033477
```

对的，上篇使用的是正则表达式实现的抓取任务专栏：004

上篇的实现还存在好多瑕疵，文本好存在好些不需要的信息。这次我们使用BeautifulSoup来实现看看。

```
# -*- coding:utf-8 -*-
# To: learn BeautifulSoup
# Date: 2016.04.29
# Author: wuxiaoshen

from bs4 import BeautifulSoup
import requests
import re
import codecs

class LiuweipengBlog(object):
    def __init__(self):
        self.url = "http://blog.csdn.net/pongba/article/details/40
33477"
        self.header = {"User-Agent": 'Mozilla/5.0 (Windows NT 6.1;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.110
Safari/537.36',
           "Referer": 'http://blog.csdn.net/pongba/article/detail
s/7911997'}

        pass

    def download(self):
        html = requests.get(self.url, headers=self.header)
        try:
            if html.status_code == 200:
                return html.content
        except:
            print("Something wrong with it.")

        pass

    def parse_content(self, content):
        Soup = BeautifulSoup(content, "lxml",from_encoding='utf-
8')
        words_of_passage = Soup.find(id="article_content")
        # print(words_of_passage)
        words = BeautifulSoup(str(words_of_passage), "lxml", from_
encoding='utf-8')
        all_words = words.find_all('p')
        with codecs.open("LiuWeiPeng.txt", "w+", encoding='utf8')
as f:
            for one in all_words:
                f.write(one.get_text())
                f.write('\n')

if __name__ == "__main__":
    Blog_passage = LiuweipengBlog()
```

```
content = Blog_passage.download()
passage = Blog_passage.parse_content(content)
```

你可能已经看出来，我只是对部分代码进行了重构。

结果部分显示截图：干净很多了。当然还是可以继续优化。继续完善。(你懂的，我不是个完美的人)
事实是，实际工程中为了得到所需要的信息，通常会混合使用这些解析方法。

```
1  你所拥有的知识并不取决于你记得多少，而在于它们能否在恰当的时候被回忆起来。
2  让我稍微说得更详细一点：学习新知识并将其存放于大脑中，最终的目的是要在恰当的时候能够想得起来去使用。
3  这可不像它听上去那么简单，否则就不会有“掉书袋”、“读死书”这种修辞手法了。
4  为了更深入地说明这一点，以下是几个著名的关于学习与记忆机制的实验：
5  《找寻逝去的自我》上提到这样一个例子：
6  假设这样一个任务：给你一个单词（如brain），要你寻找它的押韵单词（如train）。一段时间之后问你记不记得
7  对此一个靠谱的解释是：后一种记忆编码方式（称为精细编码）提供了更多的提取线索。所谓条条大路通罗马，任
8  一个非常类似的实验是这样进行的（只记了实验，忘了出处了，顺便请教知道的同学:) Update：感谢 Leeve
9  指教，这是 Craik&Tulving 于 1975 年做的一个关于记忆的浅层深层加工的经典实验，参考这里，论文原文可参
10 给出同样一组单词，让一组被试数一数每个单词有多少个音节，让另一组被试阅读单词的含义（或者设想单词可以
11 这是一个被广为认可的记忆机制，即：我们在记忆的时候将许多线索（例如当时的场景、问题的背景，甚至所处的
12 原则上，在上面提到的两个实验中，两组被试都接触到了同样的单词，都记忆了同样的知识，但取决于在记忆的时
13 联系我们日常的经验，不难注意到，死板的记忆方式和我们常说的“理解记忆”正对应了不同的编码方式。书呆子记
14 然而对于理解记忆的人来说，知识中包含了精细的概念、逻辑、一般的解题原则、通用的解题手法、背景知识、类
15 Hilbert说：“是你.……”。）
16 缺乏线索的记忆就像记忆海洋中的孤岛，虽然在那里，但是难以访问。而富含线索的记忆则是罗马，条条大路通罗
17 古希腊（或者古罗马）有一种著名的记忆法就是利用空间位置线索来辅助记忆，我曾经用过类似的手法，在小规模
```

5：参考及总结

参考文献列表：

- BeautifulSoup文档中文翻译版
- 专栏004：网页下载器的使用
- 爬虫系列专栏

总结：看文档。(其实我都有些忘记了…)

> 关于本人：
> 国内小硕，半路出家的IT学习者。
> 兴趣领域：爬虫 ， 数据科学
> 本人正在构建一个共同成长爬虫小型社群。有兴趣私信。
> 未来，文档及代码会托管在Github上。