**BeautifulSoup基本用法总结**

2017年02月24日 15:07:51

阅读数：20584

BeautifulSoup是Python的一个库，最主要的功能就是从网页爬取我们需要的数据。BeautifulSoup将html解析为对象进行处理，全部页面转变为字典或者数组，相对于正则表达式的方式，可以大大简化处理过程。

**0x01 安装**

建议安装BeautifulSoup 4版本 利用pip进行安装:

pip install beautifulsoup4

* 1

BeautifulSoup默认支持Python的标准HTML解析库，但是它也支持一些第三方的解析库：

| **序号** | **解析库** | **使用方法** | **优势** | **劣势** |
| --- | --- | --- | --- | --- |
| 1 | Python标准库 | BeautifulSoup(html,’html.parser’) | Python内置标准库；执行速度快 | 容错能力较差 |
| 2 | lxml HTML解析库 | BeautifulSoup(html,’lxml’) | 速度快；容错能力强 | 需要安装，需要C语言库 |
| 3 | lxml XML解析库 | BeautifulSoup(html,[‘lxml’,’xml’]) | 速度快；容错能力强；支持XML格式 | 需要C语言库 |
| 4 | htm5lib解析库 | BeautifulSoup(html,’htm5llib’) | 以浏览器方式解析，最好的容错性 | 速度慢 |

**0x02 创建对象**

导入库：

from bs4 import BeautifulSoup

* 1

创建实例：

url='http://www.baidu.com'

resp=urllib2.urlopen(url)

html=resp.read()

创建对象：

bs=BeautifulSoup(html)

* 1
* 2

格式化输出内容：

print bs.prettify()

* 1

**0x03 对象种类**

BeautifulSoup将复杂的html文档转换为树形结构，每一个节点都是一个对象，这些对象可以归纳为几种：

（1）Tag

Tag相当于html种的一个标签：

#提取Tag

print bs.title

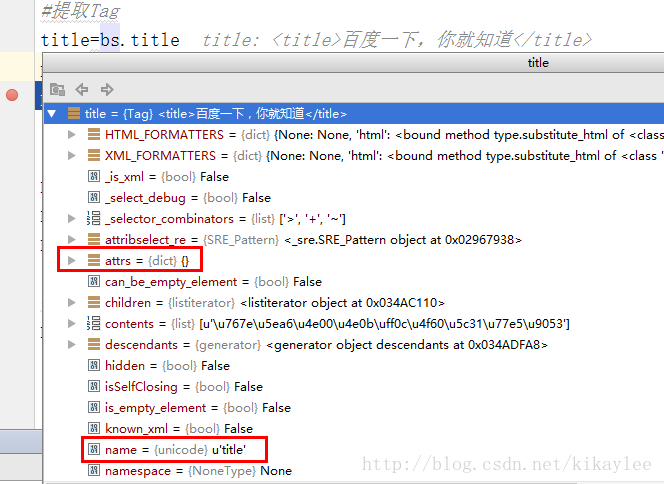
print type(bs.title)

结果：

<title>百度一下，你就知道</title>

<class 'bs4.element.Tag'>

对于Tag，有几个重要的属性：



name:每个Tag对象的name就是标签本省的名称；   
attrs:每个Tag对象的attrs就是一个字典，包含了标签的全部属性。

print bs.a.name

print bs.a.attrs

输出：

a

{u'href': u'/', u'id': u'result\_logo', u'onmousedown': u"return c({'fm':'tab','tab':'logo'})"}

（2）NavigableString

Comment是一种特殊的NavigableString，对应的是注释的内容，但是其输出不包含注释符。看这样的一个例子：

#coding:utf-8

from bs4 import BeautifulSoup

html='''

<a class="css" href="http://example.com/test" id="test"><!--test --></a>

'''

bs=BeautifulSoup(html,"html.parser")

print bs.a

print bs.a.string

* 10

运行结果：

<a class="css" href="http://example.com/test" id="test"><!--def --></a>

* 1

a标签的内容是注释，但是使用.string仍然输出了。这种情况下，我们需要做下判断：

#判断是否是注释

if type(bs.a.string)==element.Comment:

print bs.a.string

再看下面的例子：

<a class="css1" href="http://example.com/cdd" id="css">abc<!--def -->gh</a>

* 1

内容是注释和字符串混合，此时可以用contents获取全部对象：

for i in bs.a.contents:

print i

* 1
* 2

如果需要忽略注释内容的话，可以利用get\_text()或者.text：

print bs.a.get\_text()

* 1

如果想在BeautifulSoup之外使用 NavigableString 对象，需要调用unicode()方法，将该对象转换成普通的Unicode字符串，否则就算BeautifulSoup已方法已经执行结束，该对象的输出也会带有对象的引用地址，这样会浪费内存。

**0x04 搜索文档树**

重点介绍下find\_all()方法：

find\_all( name , attrs , recursive , text , \*\*kwargs )

* 1

（1）name参数

name参数可以查找所有名字为name的Tag，字符串对象自动忽略掉。

print bs.find\_all('a')

* 1

传列表：

print bs.find\_all(['a','b'])

* 1

传入正则表达式：

print bs.find\_all(re.compile('^b'))

* 1

所有以b开头的标签对象都会被找到。   
传递方法：

def has\_class\_but\_not\_id(tag):

return tag.has\_attr('class') and not tag.has\_attr('id')

print bs.find\_all(has\_class\_but\_not\_id)

* 1
* 2
* 3

（2）kwyowrds关键字

print bs.find\_all(id='css')

print bs.find\_all(id=re.compile('^a'))

* 1
* 2

还可以混合使用：

print bs.find\_all(id='css',href=re.compile('^ex'))

* 1

可以使用class作为过滤，但是class是Python中的关键字，可以使用class\_代替，或者采用字典的形式传输参数：

print bs.find\_all(class\_='css')

print bs.find\_all(attrs={'class':'css'})

* 1
* 2

（3）text参数

用来搜索文档中的字符串内容，text参数也接收字符串、正则表达式、列表、True等参数。

print bs.find\_all(text=re.compile('^abc'))

* 1

（4）limit参数

限制返回对象的个数，与数据库SQL查询类似。

（5）recursive参数

调用tag的find\_all()方法时，BeautifulSoup会检索当前tag的所有子孙节点，如果只想搜索tag的直接子节点，可以使用参数 recursive=False。

**0x05 CSS选择器**

可以采用CSS的语法格式来筛选元素：

#标签选择器

print bs.select('a')

#类名选择器

print bs.select('.css')

#id选择器

print bs.select('#css')

#属性选择器

print bs.select('a[class="css"]')

#遍历

for tag in bs.select('a'):

print tag.get\_text()

对于喜欢用CSS语法的人来说，这种方式非常方便。如果你仅仅需要CSS选择器的功能，那么直接使用 lxml 也可以，而且速度更快，支持更多的CSS选择器语法，但Beautiful Soup整合了CSS选择器的语法和自身方便使用API。