2.3 BeautifulSoup查找HTML元素

深圳信息职业技术学院
Shenzhen Institute Of Information Technology
教师：黄锐军

目 录

COMPANY

PART ONE

BeautifulSoup查找HTML元素

BeautifulSoup查找HTML元素

查找文档的元素是我们爬取网页信息的重要手段，BeautifulSoup提供了一系列的查找元素的方法，其中功能强大的find_all函数就是其中常用的一个方法。

find_all函数的原型如下：

find_all(self, name=None, attrs={}, recursive=True, text=None, limit=None, **kwargs)

self表明它是一个类成员函数；

name是要查找的tag元素名称，默认是None，如果不提供，就是查找所有的元素；

attrs是元素的属性，它是一个字典，默认是空，如果提供就是查找有这个指定属性的元素；

recursive指定查找是否在元素节点的子树下面全范围进行，默认是True；

后面的text、limit、kwargs参数比较复杂，将在后面用到时介绍；

find_all函数返回查找到的所有指定的元素的列表，每个元素是一个bs4.element.Tag对象。

find_all函数是查找所有满足要求的元素节点，如果我们只查找一个元素节点，那么可以使用find函数，它的原型如下：

find(self, name=None, attrs={}, recursive=True, text=None, limit=None, **kwargs)

使用方法与find_all类似，不同的是它只返回第一个满足要求的节点，不是一个列表。

例2-3-1：查找文档中的<title>元素

```
from bs4 import BeautifulSoup
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">Tillie</a>;
```

```
and they lived at the bottom of a well.
</p>
<p class="story">...</p>
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tag=soup.find("title")
print(type(tag),tag)
```

程序结果：

```
<class 'bs4.element.Tag'> <title>The Dormouse's story</title>
```

由此可见查找到<title>元素，元素类型是一个bs4.element.Tag对象。

例2-3-2：查找文档中的所有<a>元素

```
from bs4 import BeautifulSoup
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">Tillie</a>;
and they lived at the bottom of a well.
```

```
</p>
<p class="story">...</p>
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tags=soup.find_all("a")
for tag in tags:
    print(tag)
```

程序结果找到3个<a>元素：

```
<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>
<a class="sister" href="http://example.com/lacie"
id="link2">Lacie</a>
<a class="sister" href="http://example.com/tillie"
id="link3">Tillie</a>
```

例2-3-3：查找文档中的第一个<a>元素

```
from bs4 import BeautifulSoup
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">Tillie</a>;
and they lived at the bottom of a well.
</p>
<p class="story">...</p>
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tag=soup.find("a")
print(tag)
```

程序结果找到第一个<a>元素：

```
<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>
```

例2-3-4：查找文档中class="title"的<p>元素

```
from bs4 import BeautifulSoup
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">Tillie</a>;
```

```
and they lived at the bottom of a well.
</p>
<p class="story">...</p>
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tag=soup.find("p",attrs={"class":"title"})
print(tag)
```

程序结果找到class="title"的 元素

The Dormouse's story

很显然如果使用：

```
tag=soup.find("p")
```

也能找到这个元素，因为它是文档的第一个 元素。

例2-3-5：查找文档中class="sister"的元素

```
from bs4 import BeautifulSoup
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">Tillie</a>;
and they lived at the bottom of a well.
</p>
<p class="story">...</p>
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tags=soup.find_all(name=None,attrs={"class":"sister"})
for tag in tags:
    print(tag)
```

其中name=None表示无论是什么名字的元素，程序结果找到3个：

a class="sister" href="http://example.com/elsie" id="link1">Elsie

Lacie

Tillie

对于这个文档，很显然语句：

tags=soup.find_all("a")

或者：

tags=soup.find_all("a",attrs={"class":"sister"})

效果一样。

PART TWO

BeautifulSoup获取元素的属性值

BeautifulSoup获取元素的属性值

如果一个元素已经找到，例如找到<a>元素，那么怎么样获取它的属性值呢？BeautifulSoup使用:

tag[attrName]

来获取tag元素的名称为attrName的属性值，其中tag是一个bs4.element.Tag对象。

例2-3-6：查找文档中所有超级链接地址

```
from bs4 import BeautifulSoup
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">Tillie</a>;
and they lived at the bottom of a well.
</p>
```

```
<p class="story">...</p>
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tags=soup.find_all("a")
for tag in tags:
    print(tag["href"])
```

程序结果：

http://example.com/elsie
http://example.com/lacie
http://example.com/tillie

PART Three

BeautifulSoup获取元素包含的文本值

BeautifulSoup获取元素包含的文本值

如果一个元素已经找到，例如找到<a>元素，那么怎么样获取它包含的文本值呢？

BeautifulSoup使用: tag.text来获取tag元素包含的文本值，其中tag是一个bs4.element.Tag对象。

例2-3-7：查找文档中所有<a>超级链接包含的文本值

```
from bs4 import BeautifulSoup
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">Tillie</a>;
and they lived at the bottom of a well.
</p>
```

```
<p class="story">...</p>
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tags=soup.find_all("a")
for tag in tags:
    print(tag.text)
```

程序结果：

```
Elsie
Lacie
Tillie
```

例2-3-8：查找文档中所有<p>超级链接包含的文本值

```
from bs4 import BeautifulSoup
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">Tillie</a>;
and they lived at the bottom of a well.
</p>
<p class="story">...</p>
```

```
</body>
</html>
'''
soup=BeautifulSoup(doc,"lxml")
tags=soup.find("p")
for tag in tags:
    print(tag.text)
```

程序结果：

```
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.
...
```

其中第二个<p>包含的值就是<p>节点子树下面所有文本节点的组合值。

PART Four

BeautifulSoup高级查找

BeautifulSoup高级查找

一般find或者find_all都能满足我们的需要，如果还不能那么可以设计一个查找函数来进行查找。

例2-3-9：我们查找文档中的href="http://example.com/lacie"的节点元素<a>

```
from bs4 import BeautifulSoup
doc='''
<html><head><title>The Dormouse's story</title></head>
<body>
<a href="http://example.com/elsie" >Elsie</a>
<a href="http://example.com/lacie" >Lacie</a>
<a href="http://example.com/tillie" >Tillie</a>
```

```
</body>
</html>
'''
def myFilter(tag):
    print(tag.name)
    return (tag.name=="a" and tag.has_attr("href") and
tag["href"]=="http://example.com/lacie")

soup=BeautifulSoup(doc,"lxml")
tag=soup.find_all(myFilter)
print(tag)
```

程序结果：

html

head

title

body

a

a

a

[Lacie]

说明：

在程序中我们定义了一个筛选函数myFilter(tag)，它的参数是tag对象，在调用soup.find_all(myFilter)时程序会把每个tag元素传递给myFilter函数，由该函数决定这个tag的取舍，如果myFilter返回True就保留这个tag到结果集中，不然就丢掉这个tag。因此程序执行时可以看到html,body,head,title,body,a,a,a等一个个tag经过myFilter的筛选，只有节点Lacie满足要求，因此结果为：

[Lacie]

其中：

tag.name是tag的名称；

tag.has_attr(attName)判断tag是否有attName属性；

tag[attName]是tag的attName属性值；

例2-3-10：通过函数查找可以查找到一些复杂的节点元素，查找文本值以"cie"结尾所有<a>节点

```
from bs4 import BeautifulSoup
doc='''
<html><head><title>The Dormouse's story</title></head>
<body>
<a href="http://example.com/elsie" >Elsie</a>
<a href="http://example.com/lacie" >Lacie</a>
<a href="http://example.com/tillie" >Tillie</a>
<a href="http://example.com/tilcie" >Tilcie</a>
</body>
</html>
```

```
'''
def endsWith(s,t):
    if len(s)>=len(t):
        return s[len(s)-len(t):]==t
    return False

def myFilter(tag):
    return (tag.name=="a" and endsWith(tag.text,"cie"))

soup=BeautifulSoup(doc,"lxml")
tags=soup.find_all(myFilter)
for tag in tags:
    print(tag)
```

程序结果：

Lacie

Tilcie

程序中定义了一个endsWIth(s,t)函数判断s字符串是否以字符串t结尾，是就返回True，不然返回False，在myFilter中调用这个函数判断tag.text是否以"cie"结尾，最后找出所有文本值以"cie"结尾的<a>节点。

THANK YOU