BeautifulSoup4

> BeautifulSoup4（简称 bs4）翻译成中文就是“美丽的汤”，这个奇特的名字来源于**《爱丽丝梦游仙境》**（这也是为何在其官网会配上奇怪的插图，以及用《爱丽丝》的片段作为测试文本）。
>
> 官方文档: *https://www.crummy.com/software/BeautifulSoup/bs4/doc/index.zh.html*

安装

```
pip install Beautifulsoup4
```

使用

```
# 安装后需要在bs4中导入使用
from bs4 import BeautifulSoup

# 定义html文档内容
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">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""

# 创建一个 BeautifulSoup 对象，建议手动指定解析器:
soup = BeautifulSoup(html_doc, 'lxml')
```

tip:在使用时需要指定一个“解析器”:

html.parse- python 自带，但容错性不够高，对于一些写得不太规范的网页会丢失部分内容

lxml- 解析速度快，需额外安装 (推荐使用)

xml- 同属 lxml 库，支持 XML 文档

html5lib- 最好的容错性，但速度稍慢

一,使用Tag对象按照文档结构获取数据

```
soup.title  # title 元素
# <title>The Dormouse's story</title>
soup.p  # 第一个 p 元素
# <p class="title"><b>The Dormouse's story</b></p>
soup.p['class']  # p 元素的 class 属性
# ['title']
soup.p.b  # p 元素下的 b 元素
# <b>The Dormouse's story</b>
soup.p.parent.name  # p 元素的父节点的标签
# body
```

二,搜索

> 并不是所有信息都可以简单地通过结构化获取

使用 find 和 find_all 方法进行查找:

- find 和 find_all 可以有多个搜索条件叠加，比如**find('a', id='link3', class_='sister')**
- find 返回的是一个**bs4.element.Tag 对象**，这个对象可以进一步进行搜索。如果有多个满足的结果，find**只返回第一个**；如果没有，返回 None。
- find_all 返回的是一个**由 bs4.element.Tag 对象组成的 list**，不管找到几个或是没找到，都是 list。

```
soup.find_all('a')  # 所有 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>]

soup.find(id='link3')  # id 为 link3 的元素
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>

x = soup.find(class_='story')
x.get_text()  # 仅可见文本内容
# 'Once upon a time there were three little sisters; and their names
were\nElsie,\nLacie and\nTillie;\nand they lived at the bottom of a well.'
```

其他搜索方法

```
find_parents()          返回所有祖先节点
find_parent()           返回直接父节点
find_next_siblings()    返回后面所有的兄弟节点
find_next_sibling()     返回后面的第一个兄弟节点
find_previous_siblings() 返回前面所有的兄弟节点
find_previous_sibling()  返回前面第一个兄弟节点
find_all_next()         返回节点后所有符合条件的节点
find_next()             返回节点后第一个符合条件的节点
find_all_previous()     返回节点前所有符合条件的节点
find_previous()         返回节点前所有符合条件的节点
```

三,使用css选择器

> 如果你有前端开发经验，对 CSS 选择器很熟悉，bs 也为你提供了相应的方法：
>
> css选择器参考语法: http://www.w3school.com.cn/cssref/css_selectors.asp

```
from bs4 import BeautifulSoup
html = ''' <html> <head><title>标题</title></head> <body> <p class="title"
name="dromouse"><b>标题</b></p> <div name="divlink"> <p> <a href="http://example.com/1"
class="sister" id="link1">链接1</a> <a href="http://example.com/2" class="sister"
id="link2">链接2</a> <a href="http://example.com/3" class="sister" id="link3">链接3</a>
</p> </div> <p></p> <div name='dv2'></div> </body> </html> '''
soup = BeautifulSoup(html, 'lxml')

# 通过tag查找
print(soup.select('title'))
# [<title>标题</title>]

# 通过tag逐层查找
print(soup.select("html head title"))
# [<title>标题</title>]

# 通过class查找
print(soup.select('.sister'))
# [<a class="sister" href="http://example.com/1" id="link1">链接1</a>,
# <a class="sister" href="http://example.com/2" id="link2">链接2</a>,
# <a class="sister" href="http://example.com/3" id="link3">链接3</a>]

# 通过id查找
print(soup.select('#link1, #link2'))
# [<a class="sister" href="http://example.com/1" id="link1">链接1</a>,
# <a class="sister" href="http://example.com/2" id="link2">链接2</a>]

# 组合查找
print(soup.select('p #link1'))
# [<a class="sister" href="http://example.com/1" id="link1">链接1</a>]

# 查找直接子标签
print(soup.select("head > title"))
# [<title>标题</title>]

print(soup.select("p > #link1"))
# [<a class="sister" href="http://example.com/1" id="link1">链接1</a>]

print(soup.select("p > a:nth-of-type(2)"))
# [<a class="sister" href="http://example.com/2" id="link2">链接2</a>]

# nth-of-type 是CSS选择器

# 查找兄弟节点（向后查找）
print(soup.select("#link1 ~ .sister"))
# [<a class="sister" href="http://example.com/2" id="link2">链接2</a>,
# <a class="sister" href="http://example.com/3" id="link3">链接3</a>]

print(soup.select("#link1 + .sister"))
```

```
# [<a class="sister" href="http://example.com/2" id="link2">链接2</a>]
# 通过属性查找
print(soup.select('a[href="http://example.com/1"]'))

# ^ 以xx开头
print(soup.select('a[href^="http://example.com/"]'))

# * 包含
print(soup.select('a[href*=".com/"]'))

# 查找包含指定属性的标签
print(soup.select('[name]'))

# 查找第一个元素
print(soup.select_one(".sister"))
```