Beautiful Soup 简介

施嘉伟 180809335

一、安装

```
pip install beautifulsoup4
pip install lxml
pip install html5lib
```

二、导入

```
from bs4 import BeautifulSoup
import requests
```

三、构造方法

解析器	使用方法	优点	缺点
Python 标准库	BeautifulSoup(html, “html.parser”)	Python 的内置标准库、执行速度适中 、文档容错能力强	Python 2.7.3 or 3.2.2)前的版本中文容错能力差
lxml HTML 解析器	BeautifulSoup(html, “lxml”)	速度快、文档容错能力强	需要安装 C 语言库
Lxml XML 解析器	BeautifulSoup(html, “xml”)	速度快、唯一支持 XML 的解析器	需要安装 C 语言库
Html5lib	BeautifulSoup(html, “html5lib”)	最好的容错性、以浏览器的方式解析文档、生成 HTML5 格式的文档	速度慢、不依赖外部扩展

四、测试案例

（1）html = """ <html><head><title>The Dormouse's story</title></head> <body> 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. ... """

（2）html = """ <html><head><title>The Dormouse's story</title></head> <body> 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. ... """

（3）html = """ <html> <head> <title>The Dormouse's story</title> </head> <body> Once upon a time there were three little sisters; and their names were <a href="http://example.com/elsie"

class="sister" id="link1"> Elsie Lacie and Tillie and they lived at the bottom of a well. ... """

（4）html='''   <h4>Hello</h4>   <ul class="list" id="list-1"> <li class="element">Foo <li class="element">Bar <li class="element">Jay <ul class="list list-small" id="list-2"> <li class="element">Foo <li class="element">Bar   '''

五、选择标签（假设 **soup** 是已经构造完的 **Beautiful Soup**）

1. 选择元素

soup.title

soup.head

soup.p

2. 获取名称

soup.title.name

3. 获取属性

soup.p.attrs['name']

soup.p['name']

4. 获取内容

soup.p.string

5. 嵌套选择

soup.head.title.string

6. 子节点和子孙节点

soup.p.children（子标签）

soup.p.descendants（所有子子孙孙标签）

外：**list(enumerate())**把它转换为 **list** 可遍历数组

7. 父节点和祖父节点

soup.a.parent（父标签）

soup.a.parents（所有祖先）

8. 兄弟节点

soup.a.next_siblings

soup.a.previous_siblings

六、内置方法

1、find_all(name , attrs , recursive , text , **kwarg)

find_all 返回所有元素,可根据标签名、属性、内容查找文档

(1) name

soup.find_all('ul')

soup.find_all('ul')[0])

(2) Attrs

soup.find_all(attrs={'id': 'list-1'})

soup.find_all(attrs={'name': 'elements'})

(3) Text

soup.find_all(text='Foo')

2、find(name,attrs,recursive,text,kwargs)**

返回单个元素

contents 获取标签内所有内容

3、find_parents()

返回所有祖先节点

find_parent()

返回父节点

4、find_next_siblings()

返回后面所有兄弟节点

find_next_sibling()

返回后面第一个兄弟节点

5、find_previous_siblings()

返回前面所有兄弟节点

find_previous_sibling()

返回前面第一个兄弟节点

6、find_all_next()

返回节点后所有符合条件的节点

find_next()

返回第一个符合条件的节点

7、find_all_previous

返回节点后所有符合条件的节点

find_previous

返回第一个符合条件的节点

8、Clear

清空所有子标签（保留标签名）

soup.clear()

9、Decompose

递归删除所有标签

10、extract

递归删除所有标签，并获取标签

11、Decode

转换为字符串

decode_contents（不含当前标签转换为字符串）

12、encode

转换为字节

encode_contents（不含当前标签转换为字节）

13、has_attr

检查标签是否具有该属性

14、get_text

获取标签内部文本内容

七、来不及整理的

1. CSS 选择器（select,select_one）

2. 追加标签 (append)

3. 插入标签（insert、insert_after,insert_before）

4. 替换标签（replace_with）

5. 创建标签之间的联系

6. 包裹标签（wrap、unwrap）

7. 判断是否为空标签

七、案例演示