Python专题教程：BeautifulSoup详解

版本：v1.0

Crifan Li

摘要

本文主要介绍了Python中的第三方库，BeautifulSoup，主要用于处理HTML，此处简介什么BeautifulSoup，以及BeautifulSoup中常用的各种函数，比如find等等。

本文提供多种格式供：

在线阅读	HTML [1]	HTMLs [2]	PDF [3]	CHM [4]	TXT [5]	RTF [6]	WEBHELP [7]
下载（7zip压缩包）	HTML [8]	HTMLs [9]	PDF [10]	CHM [11]	TXT [12]	RTF [13]	WEBHELP [14]

HTML版本的在线地址为：

http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/html/python_topic_beautifulsoup.html

有任何意见，建议，提交bug等，都欢迎去讨论组发帖讨论：

http://www.crifan.com/bbs/categories/python_topic_beautifulsoup/

修订历史		
修订 1.0	2013-09-05	crl
1. 将之前在Python语言总结中的BeautifulSoup相关的内容都整理过来了 2. 把之前写的各种BeautifulSoup相关的帖子的链接也整理过来了		

[1] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/html/python_topic_beautifulsoup.html
[2] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/htmls/index.html
[3] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/pdf/python_topic_beautifulsoup.pdf
[4] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/chm/python_topic_beautifulsoup.chm
[5] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/txt/python_topic_beautifulsoup.txt
[6] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/rtf/python_topic_beautifulsoup.rtf
[7] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/webhelp/index.html
[8] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/html/python_topic_beautifulsoup.html.7z
[9] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/htmls/index.html.7z
[10] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/pdf/python_topic_beautifulsoup.pdf.7z
[11] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/chm/python_topic_beautifulsoup.chm.7z
[12] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/txt/python_topic_beautifulsoup.txt.7z
[13] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/rtf/python_topic_beautifulsoup.rtf.7z
[14] http://www.crifan.com/files/doc/docbook/python_topic_beautifulsoup/release/webhelp/python_topic_beautifulsoup.webhelp.7z

Python专题教程：BeautifulSoup详解:

Crifan Li

版本：v1.0

出版日期 2013-09-05

[15] http://www.crifan.com/files/doc/docbook/soft_dev_basic/release/html/soft_dev_basic.html#cc_by_nc

目录

前言

1. 本文目的

本文目的在于，介绍Python中第三方的，用于处理HTML（和xml）的库，BeautifulSoup

使得看完后，就明白BeautifulSoup是啥，以及基本的使用方法，

以及避免出现一些常见的错误

2. 待完成

将下面帖子内容整理合并进来：

- 【教程】Python中第三方的用于解析HTML的库：BeautifulSoup [1]
- 【总结】Python的第三方库BeautifulSoup的使用心得 [2]
- 【整理】关于Python中的html处理库函数BeautifulSoup使用注意事项 [3]
- 【已解决】BeautifulSoup已经获得了Unicode的Soup但是print出来却是乱码 [4]
- 【已解决】Python3中，已经安装了bs4（Beautifulsoup 4）了，但是却还是出错：ImportError: No module named BeautifulSoup [5]
- 【教程】BeautifulSoup中使用正则表达式去搜索多种可能的关键字 [6]
- 【整理】用BeautifulSoup查找属性值未知的标签 [7]

[1] http://www.crifan.com/python_third_party_lib_html_parser_beautifulsoup
[2] http://www.crifan.com/summary_usage_of_beautifulsoup_in_python
[3] http://www.crifan.com/some_notation_about_python_beautifulsoup_parse_html
[4] http://www.crifan.com/beautifulsoup_already_got_unicode_soup_but_print_messy_code/
[5] http://www.crifan.com/python3_after_install_bs4_still_error_importerror_no_module_named_beautifulsoup/
[6] http://www.crifan.com/python_beautifulsoup_find_using_regular_expression/
[7] http://www.crifan.com/python_use_beautifulsoup_find_tag_with_unknown_attribute_value/

第 1 章 BeautifulSoup简介

1.1. BeautifulSoup模块简介

Python的BeautifulSoup模块，可以帮助你实现HTML和XML的解析

先说一下，一般写网页爬虫，即抓取网页的html源码等内容，然后分析，提取相应的内容。

这种分析html内容的工作，如果只是用普通的正则表达式re模块去一点点匹配的话，对于内容简单点的网页分析，还是基本够用。

但是对于工作量很大，要分析的内容很繁杂的html，那么用re模块，就会发现无法实现，或很难实现。

而使用beautifulsoup模块去帮你实现分析html源码的工作的话，你就会发现，事情变得如此简单，极大地提高了分析html源码的效率。

比如我这里的想要实现博客搬家[1]之前，想要抓取对应的博客中的内容，就需要先去打开一个URL地址，去解析其中的内容，找到第一个固定链接，然后一点点分析HTML中的内容，抓去下来，导出wordpress所需要的xml文件等。

这其中对于HTML的分析，就可以利用BeautifulSoup这个模块了。

更多内容参见"Beautiful Soup 中文文档"

其中，原先链接：

http://www.crummy.com/software/BeautifulSoup/documentation.zh.html

已失效，最新的可用的地址是：

http://www.crummy.com/software/BeautifulSoup/bs3/documentation.zh.html

想要下载的话，这是BeautifulSoup的官网，其中可以下载到最新的版本：

http://www.crummy.com/software/BeautifulSoup/

[1] http://www.crifan.com/crifan_released_all/website/python/blogstowordpress/

第 2 章 BeautifulSoup中的find函数详解

2.1. BeautifulSoup中使用find，findAll等函数时，除了字符串外，也可以用正则表达式作为参数

在使用Beautifulsoup的find/finaAll等函数时候，常见用法都是传递字符串本身，比如：

```
foundIncontentTitle = lastItem.find(attrs={"class":"a-incontent a-title"});
```

可以找到：

```
<a href="/serial_story/item/7d86d17b537d643c70442326" class="a-incontent a-title"
 target=_blank>I/O-Programming_HOWTO(上)zz</a>
```

中的值。

但是却无法匹配：

```
<a href="/serial_story/item/0c450a1440b768088fbde426" class="a-incontent a-title cs-
contentblock-hoverlink" target="_blank">为什么幸运的人总幸运倒霉的人老倒霉-1　斯宾塞·约翰逊著</a>
```

即，class的值是：a-incontent a-title cs-contentblock-hoverlink，而不仅仅是a-incontent a-title

此时，如果想要匹配才class，使用传统的方法，则需要写两个find去匹配，而后来得知，原来find/findAll等函数的参数中，也可以使用正则表达式的，所以就用了：

```
titleP = re.compile("a-incontent a-title(\s+?\w+)?");# also match a-incontent a-title cs-
contentblock-hoverlink
foundIncontentTitle = lastItem.find(attrs={"class":titleP});
```

就可以一次性匹配，a-incontent a-title，a-incontent a-title cs-contentblock-hoverlink，以及未来更多可能的a-incontent a-title xxx了。

感叹一句，Beautifulsoup，做的的确很好用，特此感谢作者。

第 3 章 BeautifulSoup使用注意事项

3.1. BeautifulSoup的Tag的属性

BeautifulSoup处理这种html源码：

```
<a href="http://creativecommons.org/licenses/by/3.0/deed.zh" target="_blank">版权声明</
a>
```

后，是可以通过

```
soup.a['href']
```

去获得对应的href属性值的。

但是，想要去获得当前的某个未知的BeautifulSoup.Tag中，一共存在多少个属性，以及每个属性的值的时候，却不知道如何下手了。

比如对于如下html源码：

```
<p class="cc-lisence" style="line-height:180%;">.....</p>
```

想要得知，一共有两个属性，分别是class和style，然后就可以通过上面的方法，去获得对应的属性值了。

对此问题，虽然看到了官网的解释：Tags的属性[1]中的“你可以将Tag看成字典来访问标签的属性”

但是还是无法通过：

```
if("class" in soup)
```

的方式去判断soup中是否存在class属性，因为此时soup是Beautifulsoup.Tag类型变量，而不是dict变量。

并且，如果去强制将soup转换成为dict变量：

```
soupDict = dict(soup)
```

会报错的。

最后，还是无意间发现，原来Beautifulsoup.Tag是有个attrs属性的，其可以获得对应的元组列表，每一个元组是对应属性名和属性值：

```
attrsList = soup.attrs;
print "attrsList=",attrsList;
```

```
attrsList= [(u'class', u'cc-lisence'), (u'style', u'line-height:180%;')]
```

这样，就可以从元组列表中，自己去转换，获得属性的列表或字典变量了，就可以接着按照自己意愿去处理了。

[1] http://www.crummy.com/software/BeautifulSoup/bs3/documentation.zh.html#The%20attributes%20of%20Tags

提示

另外，此处也通过

```
soup.name
```

获得了该tag的名字

而想要获得整个soup变量所有的属性和方法的话，可以用经典的dir去打印出来:

```
print "dir(soup)=",dir(soup);
```

此处的打印输出为：

```
dir(soup)= ['BARE_AMPERSAND_OR_BRACKET',
 'XML_ENTITIES_TO_SPECIAL_CHARS', 'XML_SPECIAL_CHARS_TO_ENTITIES',
 '__call__', '__contains__', '__delitem__', '__doc__', '__eq__', '__getattr__', '__getitem__',
 '__init__', '__iter__', '__len__', '__module__', '__ne__', '__nonzero__', '__repr__',
 '__setitem__', '__str__', '__unicode__', '_convertEntities', '_findAll', '_findOne',
 '_getAttrMap', '_invert', '_lastRecursiveChild', '_sub_entity', 'append', 'attrMap',
 'attrs', 'childGenerator', 'containsSubstitutions', 'contents', 'convertHTMLEntities',
 'convertXMLEntities', 'decompose', 'escapeUnrecognizedEntities', 'extract',
 'fetch', 'fetchNextSiblings', 'fetchParents', 'fetchPrevious', 'fetchPreviousSiblings',
 'fetchText', 'find', 'findAll', 'findAllNext', 'findAllPrevious', 'findChild',
 'findChildren', 'findNext', 'findNextSibling', 'findNextSiblings', 'findParent',
 'findParents', 'findPrevious', 'findPreviousSibling', 'findPreviousSiblings',
 'first', 'firstText', 'get', 'has_key', 'hidden', 'insert', 'isSelfClosing', 'name', 'next',
 'nextGenerator', 'nextSibling', 'nextSiblingGenerator', 'parent', 'parentGenerator',
 'parserClass', 'prettify', 'previous', 'previousGenerator', 'previousSibling',
 'previousSiblingGenerator', 'recursiveChildGenerator', 'renderContents',
 'replaceWith', 'setup', 'substituteEncoding', 'toEncoding']
```

有需要的话，可以对这些属性和方法，都尝试一下，以便更加清楚其含义。

刚写完上面这句话呢，然后自己随便测试了一下attrMap：

```
attrMap = soup.attrMap;
print "attrMap=",attrMap;
```

然后就很惊喜的发现，原来此处的attrMap，就是我程序中所需要的属性的dict变量啊：

```
attrMap= {u'style': u'line-height:180%;', u'class': u'cc-lisence'}
```

这样，就又省去了我程序中将attrs转换为dict变量的操作了，更加提高了效率。在次感谢Beautifulsoup的开发者。

3.2. BeautifulSoup有时候会遇到非法的，不支持的html源码而导致无法解析或无法正常解析html

在使用Beautifulsoup过程中，对于大多数html源码，通过指定正确的编码，或者本身是默认UTF-8编码而无需指定编码类型，其都可以正确解析html源码，得到对应的soup变量。

然后就接着去利用soup实现你所想要的功能了。

但是有时候会发现，有些html解析后，有些标签等内容丢失了，即所得到的soup不是所期望的完整的html的内容。

这时候，很可能遇到了非法的html，即其中可能包含了一些不合法的html标签等内容，导致Beautifulsoup虽然可以解析，没有报错，但是实际上得到的soup变量，内容缺失了一部分了。

比如我就遇到过不少这样的例子：

1. 部分Blogbus的帖子的html中非html5和html5的代码混合导致Beautifulsoup解析错误
之前在为BlogsToWordPress添加Blogbus支持[2]过程中去解析Blogbus的帖子的时候，遇到一个特殊的帖子：http://ronghuihou.blogbus.com/logs/89099700.html，其中一堆的非html5的代码中，包含了这样一段html5的代码的写法，即标签属性值不加括号的：

```
<SCRIPT language=JavaScript>
document.oncontextmenu=new Function("event.returnValue=false;"); //禁止右键功能,单击右键将无任何反应
document.onselectstart=new Function( "event.returnValue=false;"); //禁止先择,也就是无法复制
</SCRIPT language=JavaScript>
```

结果导致Beautifulsoup解析错误，得到的soup中，找不到所需要的各种class等属性值。

对应的解决办法就是，把这部分的代码删除掉，然后再解析就可以了：

其中一堆的非html5的代码中，包含了这样一段html5的代码的写法，即标签属性值不加括号的：

```
foundInvliadScript = re.search("<SCRIPT language=JavaScript>.+</SCRIPT
 language=JavaScript>", html, re.I | re.S );
logging.debug("foundInvliadScript=%s", foundInvliadScript);
if(foundInvliadScript):
    invalidScriptStr = foundInvliadScript.group(0);
    logging.debug("invalidScriptStr=%s", invalidScriptStr);
    html = html.replace(invalidScriptStr, "");
    logging.debug("filter out invalid script OK");

soup = htmlToSoup(html);
```

2. 判断浏览器版本的相关代码，导致Beautifulsoup解析不正常
之前在给BlogsToWordpress[3]添加新浪博客的支持的过程中

遇到很多新浪博客的帖子的html中，包含很多判断浏览器版本的相关代码：

```
<!–[if lte IE 6]>
xxx
```

[2] http://www.crifan.com/record_add_blogbus_for_blogstowordpress/
[3] http://www.crifan.com/crifan_released_all/website/python/blogstowordpress/

```
xxx
<![endif]–>
```

由此导致Beautifulsoup解析html不正常。

3. font标签嵌套层次太多，导致Beautifulsoup无法解析html
接上面那个解析新浪博客帖子的例子，期间又遇到另外一个问题，对于一些特殊帖子：http://blog.sina.com.cn/s/blog_5058502a01017j3j.html

   其包含特殊的好几十个font标签且是一个个嵌套的代码，导致无法Beautifulsoup无法解析html，后来把对应嵌套的font标签删除掉，才可以正常解析。

   相关python代码为：

```
# handle special case for http://blog.sina.com.cn/s/blog_5058502a01017j3j.html
processedHtml = processedHtml.replace('<font COLOR="#6D4F19"><font
 COLOR="#7AAF5A"><font COLOR="#7AAF5A"><font COLOR="#6D4F19"><font
 COLOR="#7AAF5A"><font COLOR="#7AAF5A">', "");
processedHtml = processedHtml.replace("</FONT></FONT></FONT></FONT></
FONT></FONT>", "");
```

遇到其他类似的问题，也可以去删除或替换出错代码，即可解决问题。

不过需要说明的是，很多时候，你未必很容易就找到出错的代码。

想要找到出错的代码，更多的时候，需要你一点点调试，一点点的删除看似可疑的一些html源码，然后最终才能定位到出错的代码，然后删除掉后，才可以正常工作的。

参考书目

[1] 实现博客搬家[1]

[1] http://www.crifan.com/crifan_released_all/website/python/blogstowordpress/