Xapth

基础

- 安装
- 了解
  - 全称XML Path Language，即XML路径语言
  - 最初是用来搜寻XML文档的，但是它同样适用于HTML文档的搜索

规则

- 重点规则
  - nodename 选取此节点的所有子节点
  - / 从当前节点选取直接子节点
  - // 从当前节点选取子孙节点
  - . 选取当前节点
  - .. 选取当前节点的父节点
  - @ 选取属性

最常见的一个：//title[@class="haha"],即选取所有class属性为haha且nodename为title的节点

代码

- 导包 import lxml.etree
- 读取数据，构造element对象
  - ★html = lxml.etree.HTML(text) #text是html_str
  - html的类型 <type 'lxml.etree._Element'>
- element对象转化成str
  - lxml.etree.tostring(html)
  - 但是结果是bytes类型
- 从文件中读取html
  - lxml.etree.parse(html_file_path,解析器)
  - 解析器：etree.HTMLParser()
- ★规则提取
  - html.xpath(规则) #html是上面的element对象
  - //* 提取所有节点
  - //li 所有li节点 [<Element li at 0x1359f248>, <Element li at 0x1359f208>, <Element li at 0x1359cc08>, <Element li at 0x1359c208>, <Element li at 0x1359ac88>]里面的都是element元素，还可以继续调用xpath方法进行提取元素
  - //li/@class 取出li的所有结点的class的值，class是属于li的
    - 若其中的属性不存在，则跳过不显示
  - ★//li/a/@href 最常用的提取链接
  - //li/a/@href=\"link3.html\" 提取li标签下的a标签中href属性为link3.html的元素
  - //li//@class 取出li的所有结点的class的值，class是不一定属于li的，也可能是li下其他标签的
  - ★位置
    - //li[1] 所有li标签中的第一个
    - //li[last()] 所有li标签中的最后一个
    - //li[last()-1] 所有li标签中的倒数第二个
  - //li[last()-1]/a/@href 所有li中倒数第二个下的a标签的href属性
  - //*[@href=\"link4.html\"] 所有标签中href属性为link4.html
  - 文本
    - //*[@href=\"link4.html\"]/text() 所有标签中href属性为link4.html的标签之间的文本内容
    - ★//div[@class=\"haha\"]//p//text() 所有div标签中class属性为haha的所有p标签之间的文本内容
  - ★//li[contains(@class,"li")]/a/text() contains的使用
    - 第一个参数传入属性名称，第二个参数传入属性值，只要此属性包含所传入的属性值，就可以完成匹配了
  - ★//li[contains(@class, "li") and @name="item"]/a/text() 一个标签多个属性值
  - //li[position()<3]/a/text() position函数的使用
    - 位置小于3的li元素
- 节点轴选择
  - ancestor
    - //li[1]/ancestor::*
    - //li[1]/ancestor::div
  - attribute //li[1]/attribute::*
  - child //li[1]/child::a[@href="link1.html"]
  - descendant //li[1]/descendant::span
  - following //li[1]/following::*[2]
  - following-sibling //li[1]/following-sibling::*
  - 更多用法 http://www.w3school.com.cn/xpath/xpath_axes.asp
- XPath中的运算符
  - or 或
  - and 与
  - mod 计算余数
  - | 计算两个节点集
  - + 加法
  - - 减法
  - * 乘法
  - div 除法
  - = 等于
  - != 不等于
  - < 小于
  - > 大于
  - <= 小于或等于
  - >= 大于或等于
- 结合urllib2解析数据 标准代码

```
def download(url):
headers = {"User-Agent": "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0);"}
request = urllib2.Request(url,headers = headers)
response = urllib2.urlopen(request)
 data = response.read()
html = lxml.etree.HTML(data)
e_html = html.xpath("//*[@class=\"emphasis\"]/text()")
 print(e_html)

download("https://www.autohome.com.cn/166/#pvareaid=3311284")
```

response.read()可以直接放入到lxml.etree.HTML()中

补充

- etree模块可以自动修正HTML文本
- //ul/a
  - 当我们使用这种方式进行数据提取并没有提取到数据的时候，可能是a并不是ul的直接子节点，而是子孙节点，使用//ul//a 进行提取
  - 典型犯错的：//li[@class="item-0"]/text() text在a中不在li中，/提取不到
- ★经典问题
  - //li[@class="item-0"]/a/text()
  - //li[@class="item-0"]//text()
  - 区别
    - 第一个只会将a标签下面的文本数据挖掘出来
    - 第二个会将a下的所有文本数据挖掘出来，并且li下不属于a标签的其他文本也都会被挖掘出来
  - 联系 第一个挖掘的文本信息是第二个的子集

如果要想获取子孙节点内部的所有文本，可以直接用//加text()的方式，这样可以保证获取到最全面的文本信息，但是可能会夹杂一些换行符等特殊字符。如果想获取某些特定子孙节点下的所有文本，可以先选取到特定的子孙节点，然后再调用text()方法获取其内部文本，这样可以保证获取的结果是整洁的