Python读写XML文档(lxml方式)

发布时间：2017年4月6日 16:05 作者：杨仕航
分类标签： Python 阅读(16654) 评论(1)

读写xml文档可以说是python基础。前段时间一直用xml.dom的方式读写xml文档。

但用起来比较繁琐，尤其是读写文本的时候特别麻烦。寻思换一种新的方式读写xml。

这个lxml和xml.etree.ElementTree两个的操作方式看起来差不多。xml.etree.ElementTree也是读写xml文档的一种方式。

lxml要更好一些，使用更简洁。解析xml的时候，自动处理各种编码问题。而且它天生支持 XPath 1.0、XSLT 1.0、定制元素类。

不过，lxml不是Python自带的标准库。需要自己安装，如下方式安装：

```
pip install lxml
```

下面看看如何使用lxml读写xml文档。

1、读取xml文档

1）文档解析

lxml可以解析xml的字符串，使用etree.fromstring方法，如下所示：

```python
#coding:utf-8
from lxml import etree

xml_text = '<xml><head></head><body></body></xml>'
xml = etree.fromstring(xml_text)
```

lxml可以直接读取xml文件。

为了演示方便，找个xml文档作为例子，文件名为test.xml：

```xml
<?xml version="1.0" encoding="utf-8"?>
<root version="1.2" tag="test">
    <head>
        <title>test xml document</title>
    </head>
    <body>
        <items id="1">
            <source>aa</source>
            <target>AA</target>
        </items>
        <items id="2">
            <source>bb</source>
            <target>BB</target>
        </items>
        <items id="3">
            <source>cc</source>
            <target id="3t">CC<bpt id="3t1"/>cc</target>
        </items>
    </body>
</root>
```

lxml读取xml文件的代码如下所示：

```python
#coding:utf-8
from lxml import etree

xml = etree.parse('test.xml') #读取test.xml文件
```

2）获取属性

根节点root中有两个属性，我们可以通过如下方法获取根节点和其属性：

```python
#coding:utf-8
from lxml import etree

xml = etree.parse('test.xml') #读取test.xml文件
root = xml.getroot() #获取根节点

#获取属性
print(root.items()) #获取全部属性和属性值
print(root.keys())  #获取全部属性
print(root.get('version', '')) #获取具体某个属性
```

得到如下结果：

```
[('version', '1.0'), ('tag', 'test')]
['version', 'tag']
1.2
```

3）获取节点

假如我们不知道root节点下有什么节点，可以通过循环遍历。

```python
for node in root.getchildren():
    print(node.tag) #输出节点的标签名
```

得到如下结果：

```
head
body
```

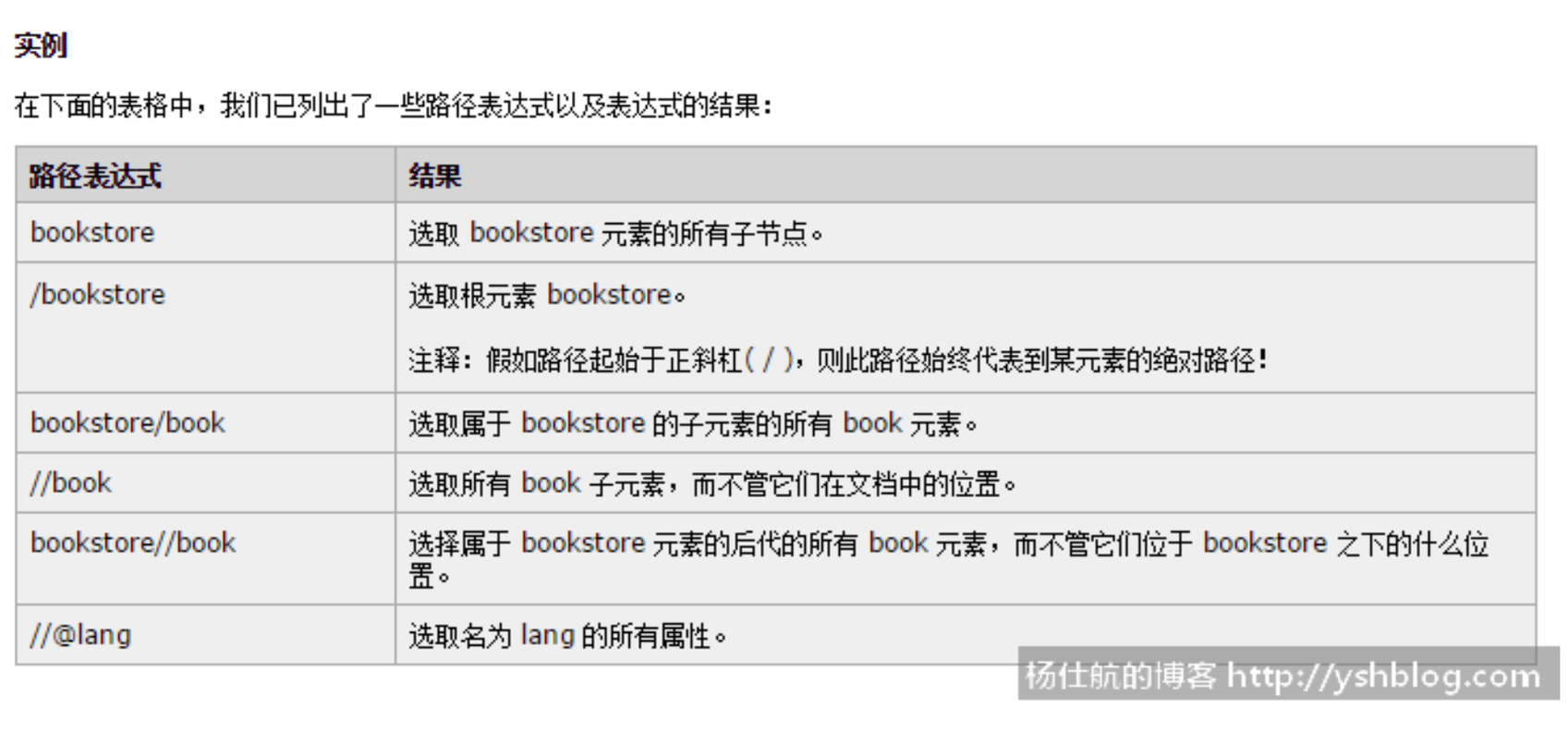
若xml文档比较大，还可以使用itergetchildren方法。该方法得到一个生成器。

这里，你可以用dir(root)可以查得节点对象有什么方法。可以获取兄弟节点、父节点等方法。

若我们知道文档结构，还可以使用XPath方法获取节点。

XPath是一种表达式，可以快速查找XML文档中的信息。可以查看w3school教程。

不要被吓到，该语法很多简单。边查阅边使用，很快可以上手。w3school中也举了一些实例：



实例

在下面的表格中，我们已列出了一些路径表达式以及表达式的结果：

路径表达式	结果
bookstore	选取 bookstore 元素的所有子节点。
/bookstore	选取根元素 bookstore。 注释：假如路径起始于正斜杠(/)，则此路径始终代表到某元素的绝对路径！
bookstore/book	选取属于 bookstore 的子元素的所有 book 元素。
//book	选取所有 book 子元素，而不管它们在文档中的位置。
bookstore//book	选择属于 bookstore 元素的后代的所有 book 元素，而不管它们位于 bookstore 之下的什么位置。
//@lang	选取名为 lang 的所有属性。

那么，我要获取全部items节点。使用XPath方法得到一个元素列表，代码如下：

```python
root.xpath('//items')
```

可得到3个items元素，无论items元素的位置在哪。

我们也可以通过root和items之间的相对路径，严格得到我们想要位置的元素：

```python
root.xpath('body/items')
```

一样可以得到3个元素，不过该xpath语法要严格很多。

假如head节点下也有items元素，该语法就不会获取到该元素。

若指定的XPath语法获取不到任何元素，将返回空列表。

4）获取文本

有些元素中有文本，这个可以通过text属性获取。

```python
#获取source元素中的文本
for node in root.xpath('//source'):
    print(node.text)
```

另外，还有一种情况，节点和文本混合的情况。

如test.xml中id为3t的target元素。target元素中有两段文本，以及文本中间还有个bpt元素。

在xml.minidom的方法，文本也是一种节点，这个问题好解决。

而在lxml中，文本不是节点。这种情况需要通过itertext方法或tail属性解决。

先尝试获取该元素的text属性：

```python
#获取id属性为3t的target元素，注意后面的[0]
target = root.xpath('//target[@id="3t"]')[0]

#输出该元素的text属性值
print(target.text)
```

将得到"CC"，后面的节点和"cc"获取不到。则text属性是获取到该节点下的第1段文本。

若该节点先是一个节点，再是文字：

```xml
<target id="3t"><bpt id="3t1"/>CCcc</target>
```

text属性将为None。

我们可以用itertext方法获取全部文本：

```python
''.join(target.itertext())
```

将得到"CCcc"。itertext方法得到一个生成器。该生成器是该节点下的全部文本生成器列表。

那假如我还需要获取其中的子节点，确保xml的结构的话，就需要使用tail属性。

tail属性是获取节点后的文本。我们可以先用text属性获取第1个文本，其他文本都通过子节点的tail属性获取。例如，我需要获取上面target元素的全部文本，若碰到子节点，则获取其id属性值一起拼成一个字符串。

```python
texts = []

#获取第1段文本
if target.text:
    texts.append(target.text)

#遍历子节点
for sub in target.iterchildren():
    texts.append('-%s-' % sub.get('id', ''))
    texts.append(sub.tail)

#合并结果
print(''.join(texts))
```

将得到"CC-3t1-cc"。

2、写入xml文档

说完读取xml文档，进入写入xml文档环节。

现我们可以模范test.xml文件，用lxml创建新的xml文档，结构和test.xml文件一致。

1）创建文档（节点）

对于lxml来说，任意节点都可以保存成一个xml文档。

我们只需要给该节点加入属性、内容、子节点等等即可。

那么创建节点方法如下：

```python
#coding:utf-8
from lxml import etree

#创建标签为root的节点
root = etree.Element('root')
```

在创建节点的同时，也可以给该节点加入命名空间：

```python
root = etree.Element('root', nsmap={'xmlns':'http://www.w3.org/1999/xhtml'})
```

在上面的test.xml中，还有两组属性。可用set方法添加属性：

```python
root.set('version', '1.2')
root.set('tag', 'test')
```

当然，也可以在创建节点的时候，就写入属性：

```python
attribs = {'version':'1.2', 'tag':'test'}
root = etree.Element('root', attrib=attribs)
```

2）添加子节点

添加根节点之后，根节点下有两个子节点：head和body。

添加子节点有两种方法，先看方法1：

```python
head = etree.Element('head')
root.append(head)
```

该方法是创建节点，再用append方法追加到root节点中。

还有一种方法，直接创建子节点：

```python
head = etree.SubElement(root, 'head')
```

推荐使用第2种方法，比较快捷。

若需要写属性值，除了用set方法。etree.SubElement方法也可以像etree.Element方法一样直接写入属性。

```python
head = etree.SubElement(root, 'head', attrib={'id':'head_id'})
```

3）添加文本

test.xml文档中，有几个地方需要添加文本。先给head添加title属性，并加入文本：

```python
title = etree.SubElement(head, 'title')
title.text = 'test xml document'
```

直接给text赋值即可。

其他文本写入我就忽略不写了，比较简单。

比较复杂的情况是上面提到的节点和文本混合的情况。这里同样给tail属性赋值即可。

```python
body = etree.SubElement(root, 'body')
items = etree.SubElement(body, 'items', attrib={'id':'3'})
target = etree.SubElement(items, 'target', attrib={'id':'3t'})

#写入第1段文本
target.text = 'CC'

#写入CC后面的节点
bpt = etree.SubElement(target, 'bpt', attrib={'id':'3t1'})

#写入第2段文本，即bpt元素后面的文本
bpt.tail = 'cc'
```

通过，该方法可以有条理的写入文本。

4）保存文档

文档写好之后，就保存文档。保存文档这里有两种方法。

一种为通过etree.tostring方法得到xml的文本，再手动写入。这个方法过于麻烦，就不讲了，也不推荐。

常规方法是通过etree的tree对象保存文件。代码如下：

```python
#节点转为tree对象
tree = etree.ElementTree(root)
tree.write('test.xml', pretty_print=True, xml_declaration=True, encoding='utf-8')
```

各个参数含义如下：

1）第1个参数是xml的完整路径(包括文件名)；

2）pretty_print参数是否美化代码；

3）xml_declaration参数是否写入xml声明，就是我们看到xml文档第1行文字；

4）encoding参数很明显是保存的编码。

lxml方式对比xml.minidom方式，有没觉得方便很多。

上一篇：Excel的日期并不特殊
下一篇：我的网站搭建(第51天) 解除评论发送邮件限制