XML

319次阅读

XML虽然比JSON复杂，在Web中应用也不如以前多了，不过仍有很多地方在用，所以，有必要了解如何操作XML。

DOM vs SAX

操作XML有两种方法：DOM和SAX。DOM会把整个XML读入内存，解析为树，因此占用内存大，解析慢，优点是可以任意遍历树的节点。SAX是流模式，边读边解析，占用内存小，解析快，缺点是我们需要自己处理事件。

正常情况下，优先考虑SAX，因为DOM实在太占内存。

在Python中使用SAX解析XML非常简洁，通常我们关心的事件是start_element，end_element和char_data，准备好这3个函数，然后就可以解析xml了。

举个例子，当SAX解析器读到一个节点时：

```
<a href="/">python</a>
```

会产生3个事件：

1. start_element事件，在读取时；
2. char_data事件，在读取python时；
3. end_element事件，在读取时。

用代码实验一下：

```
from xml.parsers.expat import ParserCreate

class DefaultSaxHandler(object):
    def start_element(self, name, attrs):
        print('sax:start_element: %s, attrs: %s' % (name, str(attrs)))

    def end_element(self, name):
        print('sax:end_element: %s' % name)

    def char_data(self, text):
        print('sax:char_data: %s' % text)

xml = r'''<?xml version="1.0"?>
<ol>
    <li><a href="/python">Python</a></li>
    <li><a href="/ruby">Ruby</a></li>
</ol>
'''
handler = DefaultSaxHandler()
parser = ParserCreate()
parser.returns_unicode = True
parser.StartElementHandler = handler.start_element
parser.EndElementHandler = handler.end_element
parser.CharacterDataHandler = handler.char_data
parser.Parse(xml)
```

当设置returns_unicode为True时，返回的所有element名称和char_data都是unicode，处理国际化更方便。

需要注意的是读取一大段字符串时，CharacterDataHandler可能被多次调用，所以需要自己保存起来，在EndElementHandler里面再合并。

除了解析XML外，如何生成XML呢？99%的情况下需要生成的XML结构都是非常简单的，因此，最简单也是最有效的生成XML的方法是拼接字符串：

```
L = []
L.append(r'<?xml version="1.0"?>')
L.append(r'<root>')
L.append(encode('some & data'))
L.append(r'</root>')
return ''.join(L)
```

如果要生成复杂的XML呢？建议你不要用XML，改成JSON。

小结

解析XML时，注意找出自己感兴趣的节点，响应事件时，把节点数据保存起来。解析完毕后，就可以处理数据。

练习一下解析Yahoo的XML格式的天气预报，获取当天和最近几天的天气：

```
http://weather.yahooapis.com/forecastrss?u=c&w=2151330
```

参数w是城市代码，要查询某个城市代码，可以在weather.yahoo.com搜索城市，浏览器地址栏的URL就包含城市代码。
