# 6.3 解析简单的XML数据¶

## 问题¶

你想从一个简单的XML文档中提取数据。

## 解决方案¶

可以使用 `xml.etree.ElementTree` 模块从简单的XML文档中提取数据。 为了演示，假设你想解析Planet Python上的RSS源。下面是相应的代码：

```
from urllib.request import urlopen
from xml.etree.ElementTree import parse

# Download the RSS feed and parse it
u = urlopen('http://planet.python.org/rss20.xml')
doc = parse(u)

# Extract and output tags of interest
for item in doc.iterfind('channel/item'):
    title = item.findtext('title')
    date = item.findtext('pubDate')
    link = item.findtext('link')

    print(title)
    print(date)
    print(link)
    print()
```

运行上面的代码，输出结果类似这样：

```
Steve Holden: Python for Data Analysis
Mon, 19 Nov 2012 02:13:51 +0000
http://holdenweb.blogspot.com/2012/11/python-for-data-analysis.html

Vasudev Ram: The Python Data model (for v2 and v3)
Sun, 18 Nov 2012 22:06:47 +0000
http://jugad2.blogspot.com/2012/11/the-python-data-model.html

Python Diary: Been playing around with Object Databases
Sun, 18 Nov 2012 20:40:29 +0000
http://www.pythondiary.com/blog/Nov.18,2012/been-...-object-databases.html

Vasudev Ram: Wakari, Scientific Python in the cloud
Sun, 18 Nov 2012 20:19:41 +0000
http://jugad2.blogspot.com/2012/11/wakari-scientific-python-in-cloud.html

Jesse Jiryu Davis: Toro: synchronization primitives for Tornado coroutines
Sun, 18 Nov 2012 20:17:49 +0000
http://feedproxy.google.com/~r/EmptysquarePython/~3/_DOZT2Kd0hQ/
```

很显然，如果你想做进一步的处理，你需要替换 `print()` 语句来完成其他有趣的事。

## 讨论¶

在很多应用程序中处理XML编码格式的数据是很常见的。 不仅因为XML在Internet上面已经被广泛应用于数据交换， 同时它也是一种存储应用程序数据的常用格式(比如字处理，音乐库等)。 接下来的讨论会先假定读者已经对XML基础比较熟悉了。

在很多情况下，当使用XML来仅仅存储数据的时候，对应的文档结构非常紧凑并且直观。 例如，上面例子中的RSS订阅源类似于下面的格式：

```
<?xml version="1.0"?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
  <channel>
    <title>Planet Python</title>
    <link>http://planet.python.org/</link>
    <language>en</language>
    <description>Planet Python - http://planet.python.org/</description>
    <item>
      <title>Steve Holden: Python for Data Analysis</title>
      <guid>http://holdenweb.blogspot.com/...-data-analysis.html</guid>
      <link>http://holdenweb.blogspot.com/...-data-analysis.html</link>
      <description>...</description>
      <pubDate>Mon, 19 Nov 2012 02:13:51 +0000</pubDate>
    </item>
    <item>
      <title>Vasudev Ram: The Python Data model (for v2 and v3)</title>
      <guid>http://jugad2.blogspot.com/...-data-model.html</guid>
      <link>http://jugad2.blogspot.com/...-data-model.html</link>
      <description>...</description>
      <pubDate>Sun, 18 Nov 2012 22:06:47 +0000</pubDate>
    </item>
    <item>
      <title>Python Diary: Been playing around with Object Databases</title>
      <guid>http://www.pythondiary.com/...-object-databases.html</guid>
      <link>http://www.pythondiary.com/...-object-databases.html</link>
      <description>...</description>
      <pubDate>Sun, 18 Nov 2012 20:40:29 +0000</pubDate>
    </item>
    ...
  </channel>
</rss>
```

`xml.etree.ElementTree.parse()` 函数解析整个XML文档并将其转换成一个文档对象。 然后，你就能使用 `find()` 、 `iterfind()` 和 `findtext()` 等方法来搜索特定的XML元素了。 这些函数的参数就是某个指定的标签名，例如 `channel/item` 或 `title` 。

每次指定某个标签时，你需要遍历整个文档结构。每次搜索操作会从一个起始元素开始进行。 同样，每次操作所指定的标签名也是起始元素的相对路径。 例如，执行 `doc.iterfind('channel/item')` 来搜索所有在 `channel` 元素下面的 `item` 元素。 `doc` 代表文档的最顶层(也就是第一级的 `rss` 元素)。 然后接下来的调用 `item.findtext()` 会从已找到的 `item` 元素位置开始搜索。

`ElementTree` 模块中的每个元素有一些重要的属性和方法，在解析的时候非常有用。 `tag` 属性包含了标签的名字， `text` 属性包含了内部的文本，而 `get()` 方法能获取属性值。例如：

```
>>> doc
<xml.etree.ElementTree.ElementTree object at 0x101339510>
>>> e = doc.find('channel/title')
>>> e
```

```
>>> e
<Element 'title' at 0x10135b310>
>>> e.tag
'title'
>>> e.text
'Planet Python'
>>> e.get('some_attribute')
>>>
```

有一点要强调的是 `xml.etree.ElementTree` 并不是XML解析的唯一方法。 对于更高级的应用程序，你需要考虑使用 `lxml` 。它使用了和ElementTree同样的编程接口，因此上面的例子同样也适用于lxml。 你只需要将刚开始的import语句换成 `from lxml.etree import parse` 就行了。 `lxml` 完全遵循XML标准，并且速度也非常快，同时还支持验证，XSLT，和XPath等特性。