6.7 利用命名空间解析XML文档¶

问题¶

你想解析某个XML文档，文档中使用了XML命名空间。

解决方案¶

考虑下面这个使用了命名空间的文档：

```
<?xml version="1.0" encoding="utf-8"?>
<top>
  <author>David Beazley</author>
  <content>
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Hello World</title>
      </head>
      <body>
        <h1>Hello World!</h1>
      </body>
    </html>
  </content>
</top>
```

如果你解析这个文档并执行普通的查询，你会发现这个并不是那么容易，因为所有步骤都变得相当的繁琐。

```
>>> # Some queries that work
>>> doc.findtext('author')
'David Beazley'
>>> doc.find('content')
<Element 'content' at 0x100776ec0>
>>> # A query involving a namespace (doesn't work)
>>> doc.find('content/html')
>>> # Works if fully qualified
>>> doc.find('content/{http://www.w3.org/1999/xhtml}html')
<Element '{http://www.w3.org/1999/xhtml}html' at 0x1007767e0>
>>> # Doesn't work
>>> doc.findtext('content/{http://www.w3.org/1999/xhtml}html/head/title')
>>> # Fully qualified
>>> doc.findtext('content/{http://www.w3.org/1999/xhtml}html/'
... '{http://www.w3.org/1999/xhtml}head/{http://www.w3.org/1999/xhtml}title')
'Hello World'
>>>
```

你可以通过将命名空间处理逻辑包装为一个工具类来简化这个过程：

```
class XMLNamespaces:
    def __init__(self, **kwargs):
        self.namespaces = {}
        for name, uri in kwargs.items():
            self.register(name, uri)
    def register(self, name, uri):
        self.namespaces[name] = '{'+uri+'}'
```

```
    def __call__(self, path):
        return path.format_map(self.namespaces)
```

通过下面的方式使用这个类：

```
>>> ns = XMLNamespaces(html='http://www.w3.org/1999/xhtml')
>>> doc.find(ns('content/{html}html'))
<Element '{http://www.w3.org/1999/xhtml}html' at 0x1007767e0>
>>> doc.findtext(ns('content/{html}html/{html}head/{html}title'))
'Hello World'
>>>
```

讨论¶

解析含有命名空间的XML文档会比较繁琐。 上面的 `XMLNamespaces` 仅仅是允许你使用缩略名代替完整的URI将其变得稍微简洁一点。

很不幸的是，在基本的 `ElementTree` 解析中没有任何途径获取命名空间的信息。 但是，如果你使用 `iterparse()` 函数的话就可以获取更多关于命名空间处理范围的信息。例如：

```
>>> from xml.etree.ElementTree import iterparse
>>> for evt, elem in iterparse('ns2.xml', ('end', 'start-ns', 'end-ns')):
... print(evt, elem)
...
end <Element 'author' at 0x10110de10>
start-ns ('', 'http://www.w3.org/1999/xhtml')
end <Element '{http://www.w3.org/1999/xhtml}title' at 0x1011131b0>
end <Element '{http://www.w3.org/1999/xhtml}head' at 0x1011130a8>
end <Element '{http://www.w3.org/1999/xhtml}h1' at 0x101113310>
end <Element '{http://www.w3.org/1999/xhtml}body' at 0x101113260>
end <Element '{http://www.w3.org/1999/xhtml}html' at 0x10110df70>
end-ns None
end <Element 'content' at 0x10110de68>
end <Element 'top' at 0x10110dd60>
>>> elem # This is the topmost element
<Element 'top' at 0x10110dd60>
>>>
```

最后一点，如果你要处理的XML文本除了要使用到其他高级XML特性外，还要使用到命名空间， 建议你最好是使用 `lxml` 函数库来代替 `ElementTree` 。 例如， `lxml` 对利用DTD验证文档、更好的XPath支持和一些其他高级XML特性等都提供了更好的支持。 这一小节其实只是教你如何让XML解析稍微简单一点。