Universidad de San Carlos de Guatemala
Maria Isabel Masaya Córdova, 201800565
Laboratorio de Introducción a la Programación 2
Sección “E”

DOM xml para Python

El Modelo de Objetos del Documento, o “DOM”, para acceder y modificar documentos XML. Una implementación del DOM presenta los documento XML como un árbol, o permite al código cliente construir dichas estructuras desde cero para luego darles acceso a la estructura a través de un conjunto de objetos que implementan interfaces conocidas.

EJEMPLO 1.

Un archivo XML simple, luego analícelo con Python minidom.

staff.xml

```xml
<?xml version="1.0"?>
<company>
    <name>Mkyong Enterprise</name>
    <staff id="1001">
        <nickname>mkyong</nickname>
        <salary>100,000</salary>
    </staff>
    <staff id="1002">
        <nickname>yflow</nickname>
        <salary>200,000</salary>
    </staff>
    <staff id="1003">
        <nickname>alex</nickname>
        <salary>20,000</salary>
    </staff>
</company>
```

Un ejemplo simple Python minidom.

Cargaremos el documento como un objeto, usando el módulo xml.dom y recuperando el objeto "minidom", que provee un acceso rápido al documento.

dom-example.py

```
from xml.dom import minidom

doc = minidom.parse("staff.xml")

# doc.getElementsByTagName returns NodeList
name = doc.getElementsByTagName("name")[0]
print(name.firstChild.data)

staffs = doc.getElementsByTagName("staff")
for staff in staffs:
        sid = staff.getAttribute("id")
        nickname = staff.getElementsByTagName("nickname")[0]
        salary = staff.getElementsByTagName("salary")[0]
        print("id:%s, nickname:%s, salary:%s" %
              (sid, nickname.firstChild.data, salary.firstChild.data))
```

Salida

```
Mkyong Enterprise
id:1001, nickname:mkyong, salary:100,000
id:1002, nickname:yflow, salary:200,000
id:1003, nickname:alex, salary:20,000
```

EJEMPLO 2

Un archivo XML simple, luego analícelo con Python minidom.

dom-example2.py

```
from xml.dom import minidom

doc = minidom.parse("staff.xml")

def getNodeText(node):

    nodelist = node.childNodes
    result = []
    for node in nodelist:
        if node.nodeType == node.TEXT_NODE:
            result.append(node.data)
    return ''.join(result)

name = doc.getElementsByTagName("name")[0]
print("Node Name : %s" % name.nodeName)
print("Node Value : %s \n" % getNodeText(name))


staffs = doc.getElementsByTagName("staff")
for staff in staffs:
        sid = staff.getAttribute("id")
        nickname = staff.getElementsByTagName("nickname")[0]
        salary = staff.getElementsByTagName("salary")[0]
        print("id:%s, nickname:%s, salary:%s" %
              (sid, getNodeText(nickname), getNodeText(salary)))
```

SALIDA

```
Node Name : name
Node Value : Mkyong Enterprise

id:1001, nickname:mkyong, salary:100,000
id:1002, nickname:yflow, salary:200,000
id:1003, nickname:alex, salary:20,000
```

Xpath (módulo python)

XPath es una sintaxis que le permite navegar a través de un xml como se usa SQL para buscar en una base de datos. Ambas funciones find y findall admiten XPath. El xml a continuación se utilizará para este ejemplo.

EJEMPLO 3

```
<Catalog>
   <Books>
      <Book id="1" price="7.95">
         <Title>Do Androids Dream of Electric Sheep?</Title>
         <Author>Philip K. Dick</Author>
      </Book>
      <Book id="5" price="5.95">
         <Title>The Colour of Magic</Title>
         <Author>Terry Pratchett</Author>
      </Book>
      <Book id="7" price="6.95">
         <Title>The Eye of The World</Title>
         <Author>Robert Jordan</Author>
      </Book>
   </Books>
</Catalog>
```

Buscando para todos los libros.

```
import xml.etree.cElementTree as ET
tree = ET.parse('sample.xml')
tree.findall('Books/Book')
```

Buscando el libro con el título 'The Colour of Magic'.

```
tree.find("Books/Book[Title='The Colour of Magic']")
# always use '' in the right side of the comparison
```

Buscando el libro con id = 5.

```
tree.find("Books/Book[@id='5']")
# searches with xml attributes must have '@' before the name
```

Buscando el segundo libro:

```
tree.find("Books/Book[2]")
# indexes starts at 1, not 0
```

Buscando el último libro:

```
tree.find("Books/Book[last()]")
# 'last' is the only xpath function allowed in ElementTree
```

Buscando todos los autores.

```
tree.findall(".//Author")
#searches with // must use a relative path
```

EJEMPLO 4

```
#!/usr/bin/python
from xml.dom.minidom import parse
import xml.dom.minidom
# Open XML document using minidom parser
DOMTree = xml.dom.minidom.parse("movies.xml")
collection = DOMTree.documentElement
if collection.hasAttribute("shelf"):
   print "Root element : %s" % collection.getAttribute("shelf")
# Get all the movies in the collection
movies = collection.getElementsByTagName("movie")
# Print detail of each movie.
for movie in movies:
print "*****Movie*****"
   if movie.hasAttribute("title"):
      print "Title: %s" % movie.getAttribute("title")
   type = movie.getElementsByTagName('type')[0]
   print "Type: %s" % type.childNodes[0].data
   format = movie.getElementsByTagName('format')[0]
   print "Format: %s" % format.childNodes[0].data
   rating = movie.getElementsByTagName('rating')[0]
   print "Rating: %s" % rating.childNodes[0].data
   description = movie.getElementsByTagName('description')[0]
   print "Description: %s" % description.childNodes[0].data
```