Nombre: Pedro Antonio Castro Calderón
201900612

DOM XML para Python

XML significa Extensible Markup Language, está diseñado para transmitir y almacenar datos, es un conjunto de reglas para definir la semántica de etiquetas. También es un lenguaje de metamarcado que define la sintaxis del lenguaje utilizado para definir otra semántica, lenguaje de marcado con estructura de dominio específico. Para poder escribir o leer documentos XML debemos instalar el paquete python3-lxml con el comando “apt-get install python3-lxml”.

Document Object Model (Document Object Model, conocido como DOM), que es un tratamiento recomendado por el W3C Lenguaje de Marcado Extensible interfaz de programación estándar.

En un analizador DOM para analizar un documento XML, se debe leer todo el documento a la vez, todos los elementos del documento deben ser guardados en una estructura de árbol en la memoria, y de esa forma puede utilizar el DOM para proporcionar diferentes funciones para leer o modificar el documento o para escribir el contenido del archivo XML.

Ejemplo 1: Analizar contenido XML con DOM

Al escribir un programa en Python, usaremos el estándar DOM del W3C. Cargaremos el documento como un objeto, usando el módulo xml.dom y recuperando el objeto "minidom", que provee un acceso rápido al documento. Tener en cuenta que tanto el fichero en Python como el fichero de ejemplo cd_catalogo.xml (descárgalo) deben estar en la misma carpeta.

Nuevo documento

```
#!/usr/bin/python
# coding: utf-8

from xml.dom.minidom import parse
import xml.dom.minidom

# Abre el documento XML usando el analizador (parser) minidom
DOMTree = xml.dom.minidom.parse("cd_catalogo.xml") #-> Modelo del Documento en forma de árbol
collection = DOMTree.documentElement # -> Objeto raiz
print "El nombre de la coleccion es: %s \n" % collection.localName

# Obtiene una lista de los objetos con la etiqueta CD
cds = collection.getElementsByTagName("CD")

```

```
11
12 # Obtiene una lista de los objetos con la etiqueta CD
13 cds = collection.getElementsByTagName("CD")
14
15 # Muestra en pantalla cada detalle de cada CD
16 for cd in cds:....
17    print "*****CD*****".
18    titulo = cd.getElementsByTagName('TITULO')[0]
19    print "Titulo: %s" % titulo.childNodes[0].data.encode("utf-8")
20    artista = cd.getElementsByTagName('ARTISTA')[0]
21    print "artista: %s" % artista.childNodes[0].data.encode("utf-8")
22    pais = cd.getElementsByTagName('PAIS')[0]
23    print "Pais: %s" % pais.childNodes[0].data.encode("utf-8")
24    comp = cd.getElementsByTagName('PAIS')[0]
25    print "Compañia: %s" % comp.childNodes[0].data.encode("utf-8")
26    precio = cd.getElementsByTagName('PRECIO')[0]
27    print "Precio: %s €" % precio.childNodes[0].data.encode("utf-8")
28    anno = cd.getElementsByTagName('ANNO')[0]
29    print "Año: %s" % anno.childNodes[0].data.encode("utf-8")
30    print "=" * 20 + "\n"
```

Ejemplo 2: Abrir fichero XML y presentarlo en Pantalla

```
+  Nuevo documento  ×  Nuevo documento
1 #!/usr/bin/python
2 # coding: utf-8
3
4 from xml.dom.minidom import parse
5 import xml.dom.minidom
6
7 # Abre el documento XML usando el analizador (parser) minidom
8 modelo = xml.dom.minidom.parse("cd_catalogo.xml") #-> Modelo # #  del Documento en forma de
  árbol. Apertura por nombre
9 # O bien
10 # fichero = open("cd_catalogo.xml")
11 # modelo = parse(fichero) # Otra forma de abrir el fichero.
12
13 coleccion = modelo.documentElement # -> Objeto raiz
14 print "El nombre de la coleccion es: %s \n" % coleccion.localName
15
16 print coleccion.toxml()
17 # print coleccion.toprettyxml() # --> formas de presentar los daros como un bloque
```

Ejemplo 3: Crear un fichero XML y guardárlo

+ Nuevo documento | Nuevo documento | × Nuevo docur

```
# coding: utf-8

from xml.dom import minidom

Ordenador1 = ['Pentium M', '512MB']
Ordenador2 = ['Pentium Core 2', '1024MB']
Ordenador3 = ['Pentium Core Duo', '1024MB']
listaOrdenadores = [Ordenador1, Ordenador2, Ordenador3]

# Abro un modelo DOM en modo implementar
DOMimpl = minidom.getDOMImplementation()

#Crear el documento econ la etiqueta principal estacionesTrabajo
xmldoc = DOMimpl.createDocument(None,"estacionesTrabajo", None)
doc_root = xmldoc.documentElement

# Recorro la lista de ordenadores
for ordenador in listaOrdenadores:
....
    #Crear Nodo... (*)
    nodo = xmldoc.createElement("Ordenador")

    # Crear un subnodo, llamado procesador
    elemento = xmldoc.createElement('Procesador')
    # Le añado un nodo de texto, y le asigno la posición 0 de la lista
    elemento.appendChild(xmldoc.createTextNode(ordenador[0]))
    # Añado el subnodo al nodo anterior
    nodo.appendChild(elemento)
....
    # Idéntico.
    elemento = xmldoc.createElement('Memoria')
    elemento.appendChild(xmldoc.createTextNode(ordenador[1]))
    nodo.appendChild(elemento)


    # (*)... que se añade como hijo al doc_root
    doc_root.appendChild(nodo)

# Recorrer para presentar en pantalla la lista de los nodos
listaNodos = doc_root.childNodes
for nodo in listaNodos:
    print nodo.toprettyxml()

# Guardar la información en un fichero de texto
fichero = open("ordenadores.xml", 'w')
# fichero.write(xmldoc.toxml())
# fichero.write(xmldoc.toprettyxml()) --> diferentes formas de guardar un fichero xml
fichero.write(xmldoc.toprettyxml(encoding="utf-8"))
fichero.close()
```

xPath XML para Python

xPath es un módulo que forma parte de la librería xml.etree.ElementTree y regularmente su importación se escribe así: “import xml.etree.ElementTree as ET”

Este módulo provee una serie de expresiones cuyo objetivo es localizar elementos en un árbol, con el fin de proporcionarun conjunto de sintaxis, pero debido a su limitado alcance no es considerado un motor en si mismo

Sintaxis de xPath

SINTAXIS	Descripción
tag	Selecciona todos los elementos hijos contenidos en la etiqueta “tag”, Por ejemplo: spam, selecciona todos los elementos hijos de la etiqueta spam y así sucesivamente en un path de nodos spam/egg, /spam/egg/milk
*****	Selecciona todos los elementos hijos. Ejemplo: */egg, seleccionara todos los elementos nietos bajo la etiqueta egg
.	Selecciona el nodo actual, este es muy usado en el inicio del path, para indicar que es un path relativo.
//	Selecciona todos los sub elementos de todos los niveles bajo el nodo expresado. Por ejemplo: .//egg selecciona todos los elementos bajo egg a través de todo el arbol bajo la etiqueta
..	Selecciona el elemento padre
[@attrib]	Selecciona todos los elementos que contienen el atributo tras el “@”
[@attrib=’value’]	Seleccione todos los elementos para los cuales el atributo dado tenga un valor dado, el valor no puede contener comillas
[tag]	Selecciona todos los elementos que contienen una etiqueta hijo llamada tag. Solo los hijos inmediatos son admitidos
[tag=’text’]	Selecciona todos los elementos que tienen una etiqueta hijo llamada tag incluyendo descendientes que sean igual al texto dado
[position]	Selecciona todos los elementos que se encuentran en la posición dada. La posición puede contener un entero siendo 1 la primera posición, la expresión last( ) para la ultima, o la posición relativa con respecto a la ultima posición last( )-1

Ejemplo 3: Encontrar nodos en un documento

Ejemplo 4: Realizando consultas xPath con lxml.

Podemos realizar consultas xpath utilizando el método xpath desde la estructura ElementTree (doc) o desde cualquier elemento seleccionado. Por ejemplo:

```
print(doc.xpath("//price/text()"))
['30.00', '29.99', '49.99', '39.95']
```

Veamos otro ejemplo, si seleccionamos todos los libros:

```
libros=doc.xpath("/bookstore/book")
```

Hemos obtenido una lista de elementos que podemos recorrer:

```
for libro in libros:
    print(libro.xpath("./title/text()")[0])
    print(libro.xpath("./@category")[0])
    print(libro.xpath(".//author/text()"))
```

En este caso hemos mostrado el título, la categoría y la lista de autores de cada libro:

```
Everyday Italian
COOKING
['Giada De Laurentiis']
Harry Potter
CHILDREN
['J K. Rowling']
XQuery Kick Start
WEB
['James McGovern', 'Per Bothner', 'Kurt Cagle', 'James Linn', 'Vaidyanathan Nagarajan']
Learning XML
WEB
['Erik T. Ray']
```