201900822 **Osmar Noel Chacón Lemus**

Dom XML en Python

Es un lenguaje que se utiliza para la creación de API´s de aplicaciones web, para acceder y modificar documentos XML. Este presenta los documentos XML como la forma de un árbol o permite al código construir estas estructuras desde cero.

DOM es útil para aplicaciones de acceso directo ya que no es obligatorio tener un árbol, DOM crea estos árboles automáticamente dando acceso a cualquier vista, parte de un documento, tener acceso a nodos, etc.

Para la creación de aplicaciones DOM para dar acceso a métodos de creación *Document* esta *DOMImplementation* y no se puede acceder a otros métodos sin un *Document* creado. En Python cada implementación del DOM proporciona una función getDOMImplementation().

En resumen, el DOM en Python se utiliza para el manejo de datos escritos en XML para guardarlos, usarlos en árboles, usarlos en nodos, retirar datos, manejar los datos extraídos para otros usos.

Ejemplos:

Se tiene un XML con datos de usuarios registrados en una empresa:

Ejemplo 1:

```
<?xml version="1.0"?>
<empresa>
<empleado id="1">
<nombre>José Ernesto</nombre>
<username>jose</username>
<password>321423</password>
</empleado>
<empleado id="2">
<nombre>Daniel Pérez</nombre>
```

```
<username>dperez</username>
<password>433543</password>
</empleado>
</empresa>
from xml.dom import minidom
doc = minidom.parse("/ruta/datos.xml")
nombre = doc.getElementsByTagName("nombre")[0]
print(nombre.firstChild.data)
empleados = doc.getElementsByTagName("empleado")
for empleado in empleados:
sid = empleado.getAttribute("id")
username = empleado.getElementsByTagName("username")[0]
password = empleado.getElementsByTagName("password")[0]
print("id:%s " % sid)
print("username:%s" % username.firstChild.data)
print("password:%s" % password.firstChild.data)
```

Ejemplo 2:

Analizar contenido XML con DOM

Se tomo como ejemplo el documento https://drive.google.com/file/d/0B7fR3YvGyQHQS0hNcldIYTJnOG8/view?usp=sharing

```
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 raíz
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")

# Muestra en pantalla cada detalle de cada CD
```

```
for cd in cds:
   print "*****CD*****"
   titulo = cd.getElementsByTagName('TITULO')[0]
   print "Título: %s" % titulo.childNodes[0].data.encode("utf-8")
   artista = cd.getElementsByTagName('ARTISTA')[0]
   print "artista: %s" % artista.childNodes[0].data.encode("utf-
8")
   pais = cd.getElementsByTagName('PAIS')[0]
   print "País: %s" % pais.childNodes[0].data.encode("utf-8")
   comp = cd.getElementsByTagName('PAIS')[0]
   print "Compañía: %s" % comp.childNodes[0].data.encode("utf-8")
   precio = cd.getElementsByTagName('PRECIO')[0]
   print "Precio: %s €" % precio.childNodes[0].data.encode("utf-
8")
   anno = cd.getElementsByTagName('ANNO')[0]
   print "Año: %s" % anno.childNodes[0].data.encode("utf-8")
   print "=" * 20 + "\n"
```

Ejercicio 3

Crear un Fichero en XML

```
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()
```

Ejercicio 4

Extraer Texto y Modificarlo

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

# Abre el documento XML usando el analizador (parser) minidom
modelo = xml.dom.minidom.parse("cd_catalogo.xml") #-> Modelo
# #  del Documento en forma de árbol. Apertura por nombre
# O bien
# fichero = open("cd_catalogo.xml")
# modelo = parse(fichero) # Otra forma de abrir el fichero.
# después al final habrá que cerrar el fichero.
fichero.close()

coleccion = modelo.documentElement # -> Objeto raíz
print "El nombre de la coleccion es: %s \n" %
coleccion.localName

cds = coleccion.getElementsByTagName("CD")

for cd in cds:

    titulo = cd.getElementsByTagName("TITULO")
    # titulo es una lista de Nodos, aunque sólo sea uno. Lo
siguiente print titulo.toxml() no funcionará
```

```
    # print titulo[0].toxml() Esto si funcionará
    # print titulo[0].childNodes # --> Esto es un objeto
    print titulo[0].childNodes[0].data #--> Accede a un dato
de texto
    # print titulo[0].childNodes[0].data.encode("utf-8") #->
no está de más codificar en utf-8
    if titulo[0].childNodes[0].data.encode("utf-8") ==
"Empire Burlesque":
        print "detectado"
        titulo[0].childNodes[0].data = u'Imperio Burlesco' #
--> Tiene que ser en formato UNICODE u''
        print titulo[0].childNodes[0].data #--> Accede a un
dato de texto
    print "=" * 20

fichero = open("cd_catalogo2.xml","w")
# fichero = open("cd_catalogo.xml","w") # --> Para
sobrescribir en el mismo fichero.
fichero.write(coleccion.toxml(encoding='utf-8')) #--> Forzar
la codificación a UTF-8
fichero.close()
# print coleccion.toprettyxml() # --> formas de presentar los
datos como un bloque '''
```

Ejercicio 5

Generar Fichero XML con DOM

```
from xml.dom import minidom, Node

doc = minidom.Document() #--> Crear un documento xml

doc.appendChild(doc.createComment("Creando documento de ejemplo XML")) #-
> Escribir comentario

book = doc.createElement('libro') # --> Crear elemento dentro del documento
doc.appendChild(book) #-> Añadirlo a la raíz del documento

title = doc.createElement('Título') # --> Crear elemento título
title.appendChild(doc.createTextNode('D. Quijote de La Mancha')) # --> Agrgar
nodo de texto
book.appendChild(title) # --> Añadir dentro del elemento book

author = doc.createElement('Autor') # --> Crear elemento Autor
```

```
book.appendChild(author) # --> Añadir al elemento libro

name = doc.createElement('Nombre y Apellidos')  # --> Crear elemento Nombre y Apellidos
author.appendChild(name) # --> Añadir dentro de Autor

firstname = doc.createElement('Nombre') # --> Crear elemento Nombre
name.appendChild(firstname)
firstname.appendChild(doc.createTextNode('Miguel'))
name.appendChild(doc.createTextNode('Texto añadido aquí'))
lastname = doc.createElement('Apellidos') # --> Crear elemento  Apellidos
name.appendChild(lastname)
lastname.appendChild(doc.createTextNode('De Cervantes Saavedra'))

chapter = doc.createElement('Capítulo')
book.appendChild(chapter)
chapter.setAttribute('number', '1')
title = doc.createElement('title')
chapter.appendChild(title)
title.appendChild(doc.createTextNode('Capítulo Primero'))

print doc.toprettyxml(indent =' ')
```

Ejercicio 6

Presentar el documento en forma de árbol

```
from xml.dom import minidom, Node

def scanNode(node, level = 0):
    msg = node.__class__.__name__
    texto=""
    if node.nodeType == Node.ELEMENT_NODE:
        msg += ", tag: " + node.tagName
    elif node.nodeType == Node.TEXT_NODE:
        texto = ": "+node.data
    print " " * level * 4, msg, texto
    if node.hasChildNodes:
        for child in node.childNodes:
            scanNode(child, level + 1)

doc = minidom.parse('cd_catalogo.xml')
scanNode(doc)
```

XPath en Python

Es un lenguaje que permite interrogar de manera sencilla a un documento XML.

Toda la potencia de XPath se pone de manifiesto cuando el desarrollador controla la formulación de sus consultas y recupera todos los casos posibles descritos por su expresión.

Ejemplo 1:

Uso de libxml2

```
import libxml2

doc = libxml2.parseFile("tst.xml")
ctxt = doc.xpathNewContext()
res = ctxt.xpathEval("//*")
if len(res) != 2:
   print "consulta xpath: tamaño incorrecto del conjunto de nodos "
   sys.exit(1)
if res[0].name != "doc" or res[1].name != "foo":
   print "  consulta xpath: valor de conjunto de nodo incorrecto "
   sys.exit(1)
doc.freeDoc()
ctxt.xpathFreeContext()
```

Ejemplo 2:

Uso de ElementTree XPath

```
from elementtree.ElementTree import ElementTree
```

```
mydoc = ElementTree(file='tst.xml')
for e in mydoc.findall('/foo/bar'):
   print e.get('title').text
```

Ejemplo 3:

Ejemplo de consulta

```
>>> element = xml.xpath('/lista/persona')
>>> len(element)
3
>>> element[0].tag
'persona'
```

Ejemplo 4:

Generando expresiones en XPath

```
>>> a   =  etree . Elemento ( "a" )
>>> b   =  etree . SubElemento ( a ,  "b" )
>>> c   =  etree . SubElemento ( a ,  "c" )
>>> d1  =  etree . SubElemento ( c ,  "d" )
>>> d2  =  etree . SubElemento ( c ,  "d"

>>> árbol  =  etree . ElementTree ( c )
>>> print ( árbol . Getpath ( d2 ))
/ c / d [2]
>>> árbol . xpath ( tree . getpath ( d2 ))  ==  [ d2 ]
Verdadero
```

Ejemplo 5:

Expresiones Regulares

```
>>> regexpNS  =  "http://exslt.org/regular-expressions"
>>> find  =  etree . XPath ( "// * [re: test (., '^ Abc $', 'i')]"
,
...                     namespaces = { 're' : regexpNS })

>>> raíz  =  etree . XML ( "<root> <a> aB </a> <b>aBc</b> </root>"
)
>>> imprimir ( buscar ( raíz ) [ 0 ] . Texto )
```

```
aBc
```