Objetos del DOM

La definición de la API del DOM para Python se proporciona como parte de la documentación del módulo xml.dom. Esta sección simplemente enumera las diferencias entre esta API y el módulo

las funciones parse y parseString es conectar un analizador sintáctico de XML con un constructor DOM que puede aceptar eventos de análisis de cualquier analizador sintáctico SAX y convertirlos en un árbol DOM. El nombre de las funciones es quizás engañoso, pero es fácil de entender cuando se comprenden las interfaces. El análisis sintáctico del documento se completará antes de que retornen estas funciones, dichas funciones simplemente no proporcionan una implementación del analizador sintáctico por si mismas.

Ejemplos DOM xml Python

Analizar contenido XML con DOM

```
#!/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 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"
```

Apertura del fichero XML y presentación en pantalla

```
#!/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
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.

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

print coleccion.toxml()
# print coleccion.toprettyxml() # --> formas de presentar los daros como
un bloque
```

Crear un fichero XML

```
# 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()
```

Análisis de los elementos de la primera etiqueta en cd_catalogo.xml

```
# coding: utf-8

from xml.dom import minidom

xmldoc = minidom.parse('cd_catalogo.xml')
print xmldoc
grammarNode = xmldoc.firstChild
print grammarNode # CATALOGO

refNode = grammarNode.childNodes[1]
print refNode #--> Nivel 1, etiqueta CD.
print refNode.childNodes  # --> Todo lo que hay bajo la etiqueta CD

pNode = refNode.childNodes[3]
print pNode #--> El elemento que está tercero. "ARTISTA"
print pNode.toxml()  # -> Impresión de dicho nodo.

print pNode.firstChild      #-> Texto que hay dentro de "ARTISTA", pero como
objeto
print pNode.firstChild.data #-> Texto extraído del objeto anterior.
```

Presentar documento XML en forma de árbol

```
# coding: utf-8

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)
```