UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS
ECONOMIA

INVESTIGACION

Nombre: Carlos Roberto Quixtán Pérez
Carné: 201901159
Fecha: 06/02/2021

DOM xml

Las aplicaciones DOM típicamente empiezan al diseccionar (parse) el XML en un DOM. Cómo esto funciona no está incluido en el DOM nivel 1, y el nivel 2 provee mejoras limitadas. Existe una clase objeto llamada DOMImplementation que da acceso a métodos de creación de Document, pero de ninguna forma da acceso a los constructores (builders) de reader/parser/Document de una forma independiente a la implementación. No hay una forma clara para acceder a estos método sin un objeto Document existente. En Python, cada implementación del DOM proporcionará una función getDOMImplementation(). El DOM de nivel 3 añade una especificación para Cargar(Load)/Guardar(Store), que define una interfaz al lector (reader), pero no está disponible aún en la librería estándar de Python.

xml.dom.minidom es una implementación mínima de la interfaz Document Object Model (Modelo de objetos del documento), con una API similar a la de otros lenguajes. Está destinada a ser más simple que una implementación completa del DOM y también significativamente más pequeña. Aquellos usuarios que aún no dominen el DOM deberían considerar usar el módulo xml.etree.ElementTree en su lugar para su procesamiento XML.

Las aplicaciones DOM suelen comenzar analizando algún XML en un DOM. Con xml.dom.minidom, esto se hace a través de las funciones de análisis sintáctico:

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

dom1 = parse('c:\\temp\\mydata.xml')  # analizar un archivo XML por nombre

datasource = open('c:\\temp\\mydata.xml')
dom2 = parse(datasource)  # analizar un archivo abierto

dom3 = parseString('<myxml>Some data<empty/> some more data</myxml>')
```

La función parse() puede tomar un nombre de archivo o un objeto de archivo previamente abierto.

xml.dom.minidom.parse(filename_or_file, parser=None, bufsize=None)
Retorna un Document a partir de la entrada dada. filename_or_file puede ser un nombre de archivo o un objeto similar a un archivo. parser, si se proporciona, debe ser un objeto de un analizador sintáctico SAX2. Esta función intercambiará el controlador de documentos del analizador sintáctico y activará el soporte con el espacio de nombres. Otras configuraciones del analizador sintáctico (como configurar un solucionador de entidades) deben haberse realizado de antemano.

Si tienes XML en una cadena de caracteres, puedes usar la función parseString() en su lugar.

También puedes crear un objeto Document invocando a un método en un objeto de la «Implementación del DOM». Puedes obtener este objeto llamando a la función getDOMImplementation() del paquete xml.dom o del módulo xml.dom.minidom. Una vez que tengas un objeto Document, puedes agregarle nodos secundarios para llenar el DOM:

```
from xml.dom.minidom import getDOMImplementation

impl = getDOMImplementation()

newdoc = impl.createDocument(None, "some_tag", None)
top_element = newdoc.documentElement
text = newdoc.createTextNode('Some textual content.')
top_element.appendChild(text)
```

Una vez que tengas un objeto del documento DOM, puedes acceder a las partes de tu documento XML a través de sus propiedades y métodos. Estas propiedades se definen en la especificación DOM. La propiedad principal del objeto del documento es documentElement. Te proporciona el elemento principal en el documento XML: el que contiene a todos los demás. Aquí hay un programa de ejemplo:

```
dom3 = parseString("<myxml>Some data</myxml>")
assert dom3.documentElement.tagName == "myxml"
```

Otro ejemplo

```
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))
```

Xpath

Xpath es un módulo que es parte de la librería xml.etree.ElementTree por lo general la misma se importa de la siguiente manera:

1 import xml.etree.ElementTree as ET

Xpath provee una serie de expresiones para localizar elementos en un árbol, su finalidad es proporcionar un conjunto de sintaxis, por lo que debido a su limitado alcance no se considera un motor en si mismo.

Ejemplos de uso de Xpath:

```
import xml.etree.ElementTree as ET

root = ET.fromstring(docxml)

# Elementos de nivel superior
root.findall(".")

# todos los hijos de neighbor o nietos de country en el nivel superior
root.findall("./country/neighbor")

# Nodos xml con name='Singapore' que sean hijos de 'year'
root.findall(".//year/..[@name='Singapore']")

# nodos 'year' que son hijos de etiquetas xml cuyo name='Singapore'
root.findall(".//*[@name='Singapore']/year")

# todos los nodos 'neighbor'que son el segundo hijo de su padre
root.findall(".//neighbor[2]")
```

SINTAXIS

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 última, o la posición relativa con respecto a la última posición last( )-1