DOM XML

Objetos del Documento para es un lenguaje API para acceder y modificar documentos XML. Una implementación del DOM presenta los documentos 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 implementaron interfaces conocidas.

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 métodos sin un objeto Document existente. En Python, cada implementación del DOM proporcionará una función getDOMImplementation()

Una vez que tengas un objeto del documento del DOM, puedes acceder a las partes de tu documento XML a través de sus propiedades y métodos. Estas propiedades están definidas en la especificación del DOM; está porción del manual describe la interpretación de la especificación en Python.

Contenido del Módulo

- xml.dom.registerDOMImplementation(name, factory)
- xml.dom.getDOMImplementation(name=None, features=())

Algunas constantes convenientes son proporcionadas:

- xml.dom.EMPTY_NAMESPACE
- xml.dom.XML_NAMESPACE
- xml.dom.XMLNS_NAMESPACE
- xml.dom.XHTML_NAMESPACE¶

Objetos en el DOM

Interfaz	Sección	Propósito
`DOMImplementation`	Objetos DOMImplementation	Interfaz para las implementaciones subyacentes.

Interfaz	Sección	Propósito
`Node`	Objetos Nodo	Interfaz base para la mayoría de objetos en un documento.
`NodeList`	Objetos NodeList	Interfaz para una secuencia de nodos.
`DocumentType`	Objetos DocumentType	Información acerca de la declaraciones necesarias para procesar un documento.
`Document`	Objetos Documento	Objeto que representa un documento entero.
`Element`	Objetos Elemento	Nodos elemento en la jerarquía del documento.
`Attr`	Objetos Atributo	Nodos de los valores de los atributos en los elementos nodo.
`Comment`	Objetos Comentario	Representación de los comentarios en el documento fuente.
`Text`	Objetos Texto y CDATASection	Nodos con contenido textual del documento.
`ProcessingInstruction`	Objetos ProcessingInstruction	Representación de instrucción del procesamiento.

Ejemplos:

datos.xml

```
<?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>
```

Importamos minidom.

```
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)
```

xPath XML

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 sí mismo.

Xpath Sintaxis:

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

Ejemplos:

```
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]")
```