Universidad de San Carlos de Guatemala
Facultad de ingeniería
Introducción a la programación y computación
Pablo Gerardo Schaart Calderon
201800951

xml.dom

El Modelo de Objetos del Documento, o «DOM» por sus siglas en inglés, es un lenguaje API del Consorcio *World Wide Web* (W3C) para acceder y modificar documentos XML. Una implementación del DOM presenta los documento 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.

El DOM es extremadamente útil para aplicaciones de acceso directo. SAX sólo te permite la vista de una parte del documento a la vez. Si estás mirando un elemento SAX, no tienes acceso a otro. Si estás viendo un nodo de texto, no tienes acceso al elemento contendor. Cuando desarrollas una aplicación SAX, necesitas registrar la posición de tu programa en el documento en algún lado de tu código. SAX no lo hace por ti. Además, desafortunadamente no podrás mirar hacia adelante (*look ahead*) en el documento XML.

Algunas aplicaciones son imposibles en un modelo orientado a eventos sin acceso a un árbol. Por supuesto que puedes construir algún tipo de árbol por tu cuenta en eventos SAX, pero el DOM te evita escribir ese código. El DOM es una representación de árbol estándar para datos XML.

El Modelo de Objetos del Documento es definido por el *W3C* en fases, o «niveles» en su terminología. El mapeado de Python de la API está basado en la recomendación del DOM nivel 2.

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.

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.

La especificación estipulada por el *W3C* define la *DOM API* para Java, ECMAScript, y OMG IDL. El mapeo de Python definido aquí está basado en gran parte en la versión IDL de la especificación, pero no se requiere el cumplimiento estricto (aunque las implementaciones son libres de soportar el mapeo estricto de IDL). Véase la sección Conformidad para una discusión detallada del mapeo de los requisitos.

Contenido del Módulo

El módulo xml.dom contiene las siguientes funciones:

xml.dom.**registerDOMImplementation**(*name*, *factory*)

Registra la función *factory* con el nombre *name*. La función fábrica (*factory*) debe retornar un objeto que implemente la interfaz DOMImplementation. La función fábrica puede retornar el mismo objeto cada vez que se llame, o uno nuevo por cada llamada, según sea apropiado para la implementación específica (e.g. si la implementación soporta algunas personalizaciones).

xml.dom.**getDOMImplementation**(*name=None*, *features=()*)

Retorna una implementación del DOM apropiada. El *name* es o bien conocido, el nombre del módulo de una implementación DOM, o None. Si no es None importa el módulo correspondiente y retorna un objecto DomImplementation si la importación tiene éxito. Si no se le pasa un nombre, y el entorno de variable PYTHON_DOM ha sido puesto, dicha variable es usada para encontrar la información de la implementación.

Si no se le pasa un nombre, examina las implementaciones disponibles para encontrar uno con el conjunto de características requeridas. Si no se encuentra ninguna implementación, levanta una excepción ImportError. La lista de características debe ser una secuencia de pares (feature,version) que son pasados al método hasFeature() en objetos disponibles de DOMImplementation.

Algunas constantes convenientes son proporcionadas:

xml.dom.**EMPTY_NAMESPACE**

El valor usado para indicar que ningún espacio de nombres es asociado con un nodo en el DOM. Se encuentra típicamente con el namespaceURI de un nodo, o usado como el parámetro *namespaceURI* para un método específico del *namespace*.

xml.dom.**XML_NAMESPACE**

El espacio de nombres de la URI asociada con el prefijo xml, como se define por Namespaces in XML (section 4).

xml.dom.**XMLNS_NAMESPACE**

El espacio de nombres del URI para declaraciones del espacio de nombres, como se define en Document Object Model (DOM) Level 2 Core Specification (section 1.1.8).

xml.dom.**XHTML_NAMESPACE**

El URI del espacio de nombres del XHTML como se define en XHTML 1.0: The Extensible HyperText Markup Language (section 3.1.1).

Además, xml.dom contiene una clase base Node y las clases de excepciones del DOM. La clase Node proporcionada por este módulo no implementa ninguno de los métodos o atributos definidos en la especificación DOM; las implementaciones del DOM concretas deben definirlas. La clase Node propuesta por este módulo sí proporciona las constantes usadas por el atributo nodeType en objetos concretos de Node; estas son localizadas dentro de la clase en vez de estar al nivel del módulo para cumplir las especificaciones del DOM.

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:

```
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:

```
1   import xml.etree.ElementTree as ET
2
3   root = ET.fromstring(docxml)
4
5   # Elementos de nivel superior
6   root.findall(".")
7
8   # todos los hijos de neighbor o nietos de country en el nivel superior
9   root.findall("./country/neighbor")
10
11  # Nodos xml con name='Singapore' que sean hijos de 'year'
12  root.findall(".//year/..[@name='Singapore']")
13
14  # nodos 'year' que son hijos de etiquetas xml cuyo name='Singapore'
15  root.findall(".//*[@name='Singapore']/year")
16
17  # todos los nodos 'neighbor'que son el segundo hijo de su padre
18  root.findall(".//neighbor[2]")
```