Laboratorio IPC 2
Sección E
Nombre: Alejandro José García Hernández
Carné: 201800939

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 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.

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.

Ejemplos:

`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

`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

`xml.dom.`**`XHTML_NAMESPACE`**

El URI del espacio de nombres del XHTML como se define en XHTML 1.0: The Extensible HyperText Markup Language

`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

XPath para Python

El lenguaje de ruta XML (XPath) es una herramienta enormemente subestimada en el mundo del web scraping y la automatización. Imagine RegEx, pero para páginas web, eso es XPath.

Cada elemento de una página web está organizado por el Modelo de objetos de documento (DOM). El DOM es una estructura en forma de árbol, donde cada elemento representa un nodo, con rutas a los nodos padre e hijo.

XPath nos ofrece un lenguaje para atravesar rápidamente este árbol. Y, como RegEx, podemos agregar lógica a nuestra selección de nodos para hacer nuestras consultas más poderosas.

Ejemplos:

find(): Devuelve el primer Element cuya etiqueta corresponda al criterio de búsqueda que le proporcionamos. Podemos utilizarlo a partir de un objeto ElementTree, o a partir de un elemento Element, por ejemplo si queremos mostrar el precio del primer libro, podemos ponerlo de alguna de las siguientes tres maneras

findall(): Devuelve una lista de objetos `Element` que coinciden con el criterio de búsqueda. Podemos también utilizarlo a partir de un objeto `ElementTree`, o a partir de un elemento `Element`. Por ejemplo si queremos obtener todos los libros y mostrar sus precios:

findtext(): Devuelve el contenido del atributo `text` del primer elemento que coincida con el criterio de búsqueda. Por ejemplo para obtener el año de edición del primer libro, lo podemos hacer de alguna de las siguientes maneras:

iterancestors(): Nos permite iterar entre los elementos ascendentes de un elemento dado, por ejemplo si queremos conocer el padre, o el abuelo de un elemento. Por ejemplo para obtener el elemento padre de un libro:

[@attrib] Selecciona todos los elementos que contienen el atributo tras el "@"