Módulo de adaptación

Master en Business Intelligence y Big Data

PROFESOR
Antonio Sarasa Cabezuelo

Escuela de organización industrial

www.eoi.es

Manipulando XML usando Python

Procesamiento de documentos XML

- Un procesador XML permite a una aplicación acceder a los contenidos de un documento XML así como detectar posibles errores.
- Hay dos enfoques para acceder a los contenidos:
  - Dirigido por eventos.
  - Manipulación del árbol

Procesamiento de documentos XML

- Dirigido por eventos
  - El documento se procesa secuencialmente, de manera que cada elemento reconocido activa un evento que puede dar lugar a una acción por parte de la aplicación.
  - SAX es un estándar para este enfoque

Procesamiento de documentos XML

- Manipulación del árbol
  - El documento se estructura como árbol de nodos a los que se puede acceder en cualquier orden.
  - DOM es un estándar para este enfoque

Procesamiento de documentos XML

- Se van a estudiar 2 herramientas de procesamiento XML:
  - Procesamiento basado en SAX
  - Procesamiento basado en DOM

SAX(Simple API for XML)

- Es una interfaz dirigida por eventos que permite leer el contenido como una secuencia de datos e interpretar las etiquetas según se van encontrando.
- Las partes del documento siempre se leen en orden desde el inicio al final.

SAX(Simple API for XML)

- No se crea ninguna estructura de datos para representar el documento, sino que sólo se analiza secuencialmente y se generan eventos denominados eventos de análisis que corresponden con el reconocimiento de partes de un documento.
- Por ejemplo cuando se encuentra el inicio de un elemento se genera un evento o cuando finaliza un elemento se genera otro evento.

SAX(Simple API for XML)

- Para gestionar los eventos se crean funciones controladoras para cada evento que se va a considerar denominadas manejadores de eventos. De esta forma cuando ocurre un evento se llama al manejador correspondiente para que realice la acción definida en el manejador.
- No es posible manipular información ya procesada, de manera que si fuera necesario entonces habría que guardarla en una estructura de datos o bien volver a llamar al procesador.

SAX(Simple API for XML)

- Por ejemplo considerar el siguiente documento XML:

  <?xml version="1.0"?>

  <doc> <par> Hola Mundo </par> </doc>

SAX(Simple API for XML)

- El procesamiento con SAX produciría la siguiente secuencia de eventos:

  inicio de documento

  inicio de elemento doc

  inicio de elemento par

  caracteres Hola mundo

  fin de elemento par

  fin de elemento doc

  fin documento

Procesamiento SAX en Python

- Para procesar usando SAX es necesario crearse un manejador propio ContentHandler como subclase de xml.sax.ContentHandler.
- El manejador gestionara las etiquetas y atributos que se deseen del documento XML que va a ser procesado.

Procesamiento SAX en Python

- El manejador proporciona un conjunto de métodos para gestionar determinados eventos que se producen en el procesamiento:
  - Los métodos startDocument y endDocument son llamadas al comienzo y al final del archivo XML.
  - Los métodos startElement(etiqueta,atributos) y endElement(etiqueta) son llamados al comienzo y al final de cada elemento. En caso de utilizar espacios de nombres se utilizarían los métodos startElementNS y endElementNS.

Procesamiento SAX en Python

- El manejador proporciona un conjunto de métodos para gestionar determinados eventos que se producen en el procesamiento:
  - El método carácter(texto) es llamado cuando se una cadena de texto.

Procesamiento SAX en Python

- Otro conjunto de métodos utilizados en el procesamiento son:
  - xml.sax.make_parser([Lista de parsers]): Crea un nuevo objeto parser. Tiene como argumento optativo una lista de parsers.
  - xml.sax.parse(archivo XML, Manejador,[ManejadorErrores]): Crea un parser SAX y lo usa para procesar el documento XML. Tiene como argumento el documento XML que va a ser procesado, el manejador de eventos, y optativamente un manejador de errores.

Procesamiento SAX en Python

- Otro conjunto de métodos utilizados en el procesamiento son:
  - xml.sax.parseString(NombreCadenaXML, Manejador,[ManejadorErrores]): Crea un parser SAX y lo usa para procesar la cadena XML dada. Tiene como argumento la cadena XML que va a ser procesada, el manejador de eventos, y optativamente un manejador de errores.

Procesamiento SAX en Python

- Vamos a crear un manejador y luego se va a usar para procesar el documento de ejemplo:
  - Hay que importar el paquete SAX: import xml.sax
  - Se crea un clase que es subclase de la clase xml.sax.ContentHandler.
  - Se definen dentro de la clase 4 métodos:
    - El método __init__
    - Los métodos startElement y endElment
    - El método characters

Procesamiento SAX en Python

- Método __init__
  - Se define como atributos de la clase las etiquetas y atributos del documento XML que se quieren gestionar.

```
def __init__(self):
    self.Datos=""
    self.titulo=""
    self.fecha=""
    self.autor=""
```

Procesamiento SAX en Python

- Método startElement
  - Se indica que acciones se quieren llevar a cabo cuando se encuentre el comienzo de un elemento. En el ejemplo se quiere capturar el isbn del libro.

```
def startElement(self,etiqueta,atributos):
    self.Datos=etiqueta
    if etiqueta=="Libro":
        print "****Libro****"
        isbn=atributos["isbn"]
        print "isbn:", isbn
```

Procesamiento SAX en Python

- Método endElement
  - Se indica que acciones se quieren llevar a cabo cuando se encuentre el final de un elemento. En el ejemplo se quiere imprimir el nombre del elemento, y el valor que contenía.

```
def endElement(self,etiqueta):
    if self.Datos=="titulo":
        print "Título:", self.titulo
    elif self.Datos=="fecha":
        print "Fecha:",self.fecha
    elif self.Datos=="autor":
        print "Autor:", self.autor
    self.Datos=""
```

Procesamiento SAX en Python

- Método characters
  - Se indica que acciones se quieren llevar cuando se encuentre contenido textual que forma parte de un elemento.En el ejemplo se quiere almacenar dicho contenido en un atributo de la clase que luego será imprimido por pantalla.

```
def characters(self,contenido):
    if self.Datos=="titulo":
        self.titulo=contenido
    elif self.Datos=="fecha":
        self.fecha=contenido
    elif self.Datos=="autor":
        self.autor=contenido
```

Procesamiento SAX en Python

- La clase completa sería:

```
import xml.sax
class ManejadorCatalogo (xml.sax.ContentHandler)
    def __init__(self):
        self.Datos=""
        self.titulo=""
        self.fecha=""
        self.autor=""

    def startElement(self,etiqueta,atributos):
        self.Datos=etiqueta
        if etiqueta=="Libro":
            print "****Libro****"
            isbn=atributos["isbn"]
            print "isbn:", isbn

    def endElement(self,etiqueta):
        if self.Datos=="titulo":
            print "Titulo:", self.titulo
        elif self.Datos=="fecha":
            print "Fecha:",self.fecha
        elif self.Datos=="autor":
            print "Autor:", self.autor
        self.Datos=""

    def characters(self,contenido):
        if self.Datos=="titulo":
            self.titulo=contenido
        elif self.Datos=="fecha":
            self.fecha=contenido
        elif self.Datos=="autor":
            self.autor=contenido
```

Procesamiento SAX en Python

- Una vez que se tiene definida la clase, se puede llevar a cabo el procesamiento:
  - Se crea un objeto parser XML.
  - Se configura el parser.
  - Se fija el manejador de eventos.
  - Se procesa el documento.

```
parser=xml.sax.make_parser()
parser.setFeature(xml.sax.handler.feature_namespaces,0)
Handler=ManejadorCatalogo()
parser.setContentHandler(Handler)
parser.parse("Catalogo.xml")
```

Procesamiento SAX en Python

- El resultado es el siguiente:

```
****Libro****
isbn: 0-596-00128-2
Titulo: Python y XML
Fecha: Diciembre 2001
Autor: Pepito Perez
****Libro****
isbn: 0-596-15810-6
Titulo: Programacion avanzada de XML
Fecha: Octoubre 2010
Autor: Juan Garcia
****Libro****
isbn: 0-596-15806-8
Titulo: Aprendiendo Java
Fecha: Septiembre 2009
Autor: Juan Garcia
****Libro****
isbn: 0-596-15808-4
Titulo: Python para moviles
Fecha: Octubre 2009
Autor: Pepito Perez
```

DOM(Document Object Model)

- Es una especificación que proporciona una API que representa los documentos como un árbol de objetos nodo en las que hay nodos con hijos y otros nodos sin hijos denominados nodos hoja.
- Toda la jerarquía de nodos se carga en memoria de forma que se pueda recorrer y acceder a los nodos comenzando desde la raíz.

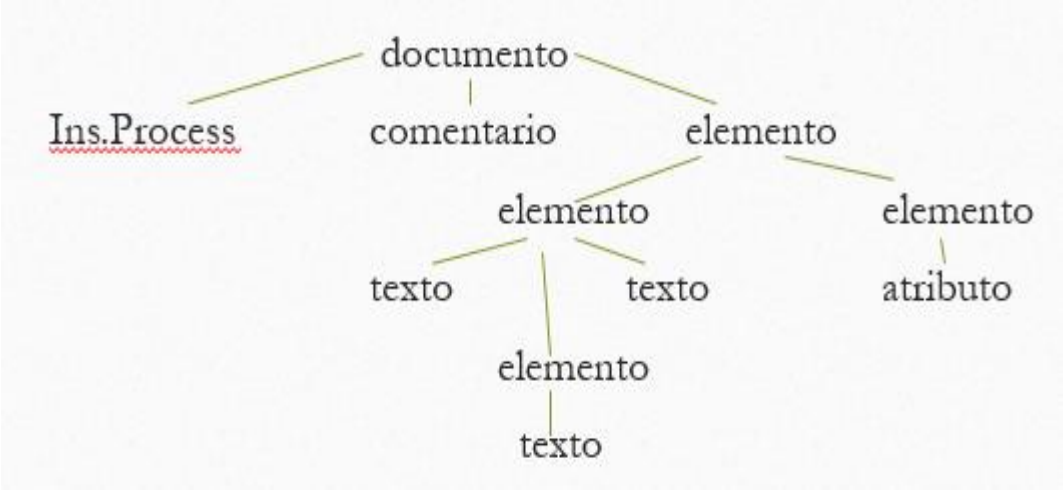
DOM(Document Object Model)

- Por ejemplo considerar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-16"?>
<doc>
    <saludo> Hola <enfatico> estimados </enfatico>
oyentes</saludo>
    <aplausos tipo="sostenido"/>
</doc>
```

DOM(Document Object Model)

- Su procesamiento daría lugar a la siguiente jerarquía de nodos:



Procesamiento DOM en Python

- Para procesar un documento XML usando DOM se debe utilizar la librería xml.dom.
- Esta librería permite crear un objeto minidom que dispone de un método que procesa un documento XML dado y genera un árbol DOM.

Procesamiento DOM en Python

- En primer lugar se abre el documento XML con el método parse del objeto minidom que proporciona un árbol DOM del documento.
- A continuación se puede empezar a recorrer el árbol. En primer lugar se accede a la raíz del árbol a través del atributo documentElement.

Procesamiento DOM en Python

- Desde la raíz del árbol se puede visitar utilizando un conjunto de métodos:
  - getElementsByTagName(Elemento): Devuelve una lista de todos los elementos que tienen el nombre proporcionado.
  - getAttribute(Atributo): Devuelve el valor del atributo proporcionado como parámetro.
  - hasAttribute(Atributo): Indica si un elemento tiene el atributo proporcionado como parámetro.
- Para acceder al contenido de cada elemento se utiliza el atributo data del objeto childNodes

Procesamiento DOM en Python

- Se va a realizar el procesamiento del documento XML de ejemplo usando DOM.

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

ArbolDOM=xml.dom.minidom.parse("Catalogo.xml")
catalogo=ArbolDOM.documentElement
libros=catalogo.getElementsByTagName("Libro")
for libro in libros:
    print "***Libro***"
    if libro.hasAttribute("isbn"):
        print "isbn:",libro.getAttribute("isbn")
    Titulo=libro.getElementsByTagName("titulo")[0]
    print "Titulo:",Titulo.childNodes[0].data
    Fecha=libro.getElementsByTagName("fecha")[0]
    print "Fecha:",Fecha.childNodes[0].data
    Autor=libro.getElementsByTagName("autor")[0]
    print "Titulo:",Autor.childNodes[0].data
```

Procesamiento DOM en Python

- Y se obtiene una salida similar:

```
***Libro***
isbn: 0-596-00128-2
Titulo: Python y XML
Fecha: Diciembre 2001
Titulo: Pepito Perez
***Libro***
isbn: 0-596-15810-6
Titulo: Programacion avanzada de XML
Fecha: Octoubre 2010
Titulo: Juan Garcia
***Libro***
isbn: 0-596-15806-8
Titulo: Aprendiendo Java
Fecha: Septiembre 2009
Titulo: Juan Garcia
```

SAX vs DOM

- SAX es un procesador bastante eficiente que permite manejar documentos muy extensos en tiempo lineal y con una cantidad de memoria constante. Sin embargo requiere de un esfuerzo mayor por parte de los desarrolladores.
- DOM es más fácil de usar para los desarrolladores pero aumenta el coste de memoria y tiempo.

SAX vs DOM

- Será mejor usar SAX cuando el documento a procesar no quepa en memoria o cuando las tareas son irrelevantes con respecto a la estructura del documento (contar el número de elementos, extraer contenido de un elemento determinado)