Curso Python en 8 clases

Clase 7: XML

Autor: Sebastián Bassi sbassi@Gmail.com
Versión: 1.1

XML

Introducción

XML es un formato de intercambio de datos. Actualmente hay todo tipo de información en este formato, se puede decir que es el formato estándar de intercambio de datos. Ejemplos de archivos XML:

- Archivos de configuración (.xml)
- Bases de datos
- Páginas web (.xhtml)
- Planillas de cálculo (.ods, .xlsx)
- Gráficos vectorizados (.svg)

Estructura de un documento XML

No hace falta conocer todas y cada una de las particularidades de los componentes de XML debido a que desde Python usaremos un parser específico para extraer los elementos deseados. De todas maneras hay que conocer un mínimo de la estructura interna para sacar el máximo provecho del parser y solucionar eventuales problemas que pueden presentarse.

> ***¡Nota importante!***
>
> Por especificación de XML, si un documento no cumple correctamente con la especificación, el parser está obligado a interrumpir su ejecución. Si un parser no lee un documento no válido, no es un error del parser, sino que está cumpliendo con la especificación. Para parsear XML no válidos, usar otra herramienta como BeautifulSoup.

Ejemplos de documentos XML

Feed de rss: Este tipo de xml está asociada a todas las páginas que llevan el logo de “RSS” y proveen este mecanismo de notificación de actualización de contenido:

```
<?xml version="1.0" ?>
<rss version="2.0">
  <channel>
    <title>miRNAdb updates</title>
    <link>http://www.misolrna.org</link>
    <description>Updates on the miRNAdb</description>
   <item>
       <title>Version 2</title>
```

```
      <link>http://www.misolrna.org</link>
      <description>Version 2 of the data uploaded. Now with more micros</description>
    </item>
   <item>
      <title>New SGN</title>
      <link>http://www.misolrna.org</link>
      <description>SGN data uploaded!</description>
    </item>
 </channel>
</rss>
```

Secuencia proteíca de Uniprot: Uno de los formatos de los registros de la base de datos de proteinas (Uniprot) es XML (el siguiente archivo fue acortado brevemente por razones de espacio):

```
<?xml version="1.0" encoding="UTF-8"?>
<uniprot xmlns="http://uniprot.org/uniprot"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://uniprot.org/uniprot
 http://www.uniprot.org/support/docs/uniprot.xsd">
<entry dataset="TrEMBL" created="2000-10-01" version="35">
  <accession>Q9JJE1</accession>
  <organism key="1">
    <name type="scientific">Mus musculus</name>
    <lineage>
      <taxon>Eukaryota</taxon>
      <taxon>Metazoa</taxon>
      <taxon>Chordata</taxon>
      <taxon>Craniata</taxon>
      <taxon>Vertebrata</taxon>
      <taxon>Euteleostomi</taxon>
      <taxon>Mammalia</taxon>
      <taxon>Eutheria</taxon>
      <taxon>Euarchontoglires</taxon>
      <taxon>Glires</taxon>
      <taxon>Rodentia</taxon>
      <taxon>Sciurognathi</taxon>
      <taxon>Muroidea</taxon>
      <taxon>Muridae</taxon>
      <taxon>Murinae</taxon>
      <taxon>Mus</taxon>
    </lineage>
  </organism>
  <dbReference type="UniGene" id="Mm.248907" key="5"/>
  <sequence length="393" checksum="E0C0CC2E1F189B8A">
MPKKKPTPIQLNPAPDGSAVNGTSSAETNLEALQKKLEELELDEQQRKRL
EAFLTQKQKVGELKDDDFEKISELGAGNGGVVFKVSHKPSGLVMARKLIH
LEIKPAIRNQIIRELQVLHECNSPYIVGFYGAFYSDGEISICMEHMDGGS
LDQVLKKAGRIPEQILGKVSIAVIKGLTYLREKHKIMHRDVKPSNILVNS
RGEIKLCDFGVSGQLIDSMANSFVGTRSYMSPERLQGTHYSVQSDIWSMG
LSLVEMAVGRYPIPPPDAKELELLFGCHVEGDAAETPPRPRTPGGPLSSY
GMDSRPPMAIFELLDYIVNEPPPKLPSGVFSLEFQDFVNKCLIKNPAERA
DLKQLMVHAFIKRSDAEEVDFAGWLCSTIGLNQPSTPTHAASI
</sequence>
</entry>
</uniprot>
```

Elementos de XML

En general los archivos XML tienen *prólogo*, *cuerpo* y (a veces), *epílogo*. El prólogo puede tener una o varias lineas:

Prólogo de una línea:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Prologo de varias líneas:

```
<?xml version="1.0"?>
<!DOCTYPE BlastOutput PUBLIC "-//NCBI//NCBI BlastOutput/EN"
"http://www.ncbi.nlm.nih.gov/dtd/NCBI_BlastOutput.dtd">
<!-- edited with XMLSPY (http://www.xmlspy.com) by Andy -->
```

En el cuerpo residen los elementos, como por ejemplo:

```
<taxon>Eukaryota</taxon>
```

Donde *<taxon>* es la **etiqueta de inicio**, *</taxon>* la **etiqueta de final** (o cierre) y *Eukaryota* es el **contenido**. Los elementos pueden estar vacios:

```
<accession></accession>
```

Esto es válido para el formato XML, pero no lo es necesariamente para el validador particular de la aplicación (sería posible que un programa de base de datos pida que el campo accession sea obligatorio, esto depende de cada aplicación particular).

Otra manera de indicar campos vacios:

```
<accession />
```

Algunos elementos pueden tener **atributos** con un **valor asociado**:

```
<elemento nombre_del_atributo="valor">
```

Como por ejemplo:

```
<name type="scientific">
```

El **elemento** *name* tiene el **atributo** *type* cuyo **valor** es *scientific*. Al diseñar documentos XML no siempre es claro cuando incluir información como valor de un atributo o como un contenido de un elemento.

Métodos de acceso a un documento XML

- **SAX**: Acceso activado por eventos, se puede ir procesando a medida que se lee el documento.
- **DOM**: Se hace una representación tipo árbol que requiere leer todo el documento de antemano.

Python soporta ambos modelos.

Parsers SAX en Python: xml.sax, cElementTree.Iterparse

Parsers DOM en Python: xml.dom, cElementTree

Ejemplo de uso con cElementTree

Búsqueda de un elemento

```
import xml.etree.cElementTree as cET
tree = cET.parse('Mixml.XML')
root = tree.getroot()
cod_t = root.find('DetalleReceta').find('Item').find('CodTroquel').text
print cod_t
```

Otra opción:

```
cod_t = root.find('DetalleReceta/Item/CodTroquel').text
```

Modificando un elemento

El siguiente código modifica el valor asociado al elemento *ImporteUnitario*, sumandole un 10%. Note el uso de *urllib2* para bajar el archivo XML desde un URL (dirección de Internet).

```
import urllib2
import xml.etree.cElementTree as cET

url = 'http://www.genesdigitales.com/curso/Mixml2.XML'
fin = urllib2.urlopen(url)
tree = cET.parse(fin)
root = tree.getroot()
for imp in root.getiterator("ImporteUnitario"):
    imp.text = '%.2f' %(float(imp.text) + float(imp.text)*.1)
```

Escribiendo el XML modificado

Una vez modificado un XML, es cuestión de usar el método write:

```
tree.write("aumentado.xml", encoding="iso-8859-1")
```

Hay que notar que el archivo resultante es un XML válido y respeta la estructura almacenada aunque puede tener un formato distinto al original.

XML con Beautiful Soup

Beautiful Soup no es parte de la biblioteca estándar, por lo que debe bajarse e instalarse por separado:

```
$ sudo easy_install BeautifulSoup
```

Este módulo permite procesar archivos que no están completos (por ejemplo les falta cerrar una etiqueta) o que contienen errores (como etiquetas mal aninadas). Por eso se lo usa mucho con documentos HTML (que suelen tener errores de todo tipo).

Hay dos métodos principales: BeautifulSoup y BeautifulStoneSoup. El primero está optimizado para documentos HTML mientras que BeautifulStoneSoup sirve para procesar documentos XML. La misma búsqueda que hicimos con **cElementTree**, se puede hacer de esta manera con **BeautifulStoneSoup**:

```
from BeautifulSoup import BeautifulStoneSoup as bs
soup = bs(open('Mixml.XML'))
cod_t = soup.detallereceta.item.codtroquel.string
print cod_t
```

Para modificar un elemento:

```
from BeautifulSoup import BeautifulStoneSoup as bs
soup = bs(open('Mixml2.XML'))
for imp in soup.findAll("importeunitario"):
    imp.string.replaceWith('%.2f' %(float(imp.string) + float(imp.string)*.1))
```

Escribiendo el XML:

```
outf = open("aumentado.xml",'w')
outf.write(soup.prettify(encoding='utf-8'))
outf.close()
```

Crear un documento desde 0

El siguiente código usa *dom* para hacer un documento XML:

```
from xml.dom.minidom import Document
# Crear un documento minidom
doc = Document()
# Crear el elemento base (<wml>)
wml = doc.createElement("wml")
doc.appendChild(wml)
# Crear el elemento <card> con atributos
maincard = doc.createElement("card")
maincard.setAttribute("id", "main")
wml.appendChild(maincard)
# Crear elemento <p>
paragraph1 = doc.createElement("p")
maincard.appendChild(paragraph1)
# Darle un texto al elemento <p>
ptext = doc.createTextNode("Esto es una prueba!")
paragraph1.appendChild(ptext)
# Imprimir el XML recien creado
print doc.toprettyxml(indent="  ")
```

(Fuente: http://www.postneo.com/projects/pyxml/)

Resultado:

```
<?xml version="1.0" ?>
<wml>
  <card id="main">
    <p>
      Esto es una prueba!
    </p>
  </card>
</wml>
```

Mas información

Element tree: http://docs.python.org/library/xml.etree.elementtree.html

Extensible Markup Language (XML). Links to W3C Recommendations, Proposed Recommendations and Working Drafts: http://www.w3.org/XML

Software Carpentry course, by Greg Wilson. XML: http://swc.scipy.org/lec/xml.html

Mark Pilgrim. Dive into Python. XML Processing: http://diveintopython.org/xml_processing

Python and XML: An Introduction: http://www.boddie.org.uk/python/XML_intro.html

Resources on XML Schema: http://www.w3schools.com/schema/schema_intro.asp

Tutorial Minidom: http://sites.google.com/site/sbassi/leyendoxmlenpython:dom2

BeautifulSoup: http://www.crummy.com/software/BeautifulSoup/

Agradecimientos

- Gabriel Genellina: Beautiful Soup.