Diseño de Aplicaciones para Internet (2014-2015)
Guión de Prácticas 4(a):
Utilización de Servicios On-line: RSS

S. Alonso, J.M. Guirao
zerjioi@ugr.es, jmguirao@ugr.es

Resumen

Una vez que dominamos los rudimentos de la programación web con `web.py` vamos a aprender a utilizar servicios que están disponibles e intentar integrarlos en nuestro sitio web. En esta primera práctica vamos a manejar la sindicación de noticias: `RSS` (Really Simple Syndication), un formato basado en `XML` utilizado para compartir información actualizada frecuentemente.

1. Introducción al `RSS`

Consulte las siguientes referencias[1, 2] (ignorando las secciones que se refieren a `PHP`) para tener una idea más o menos clara del funcionamiento de esta tecnología.

Busque por Internet páginas que ofrezcan contenidos sindicados con `RSS`. Compruebe como visualiza su navegador dichos contenidos y examine el código fuente (`XML`) de los mismos. Algunos sitios donde mirar:

- Web de El País: `http://elpais.es`
- Meneame: `http://meneame.net`
- Barrapunto: `http://barrapunto.com`

Descargue alguno de los ficheros `RSS` a su ordenador para poder trabajar cómodamente con ellos.

2. Analizando RSS

2.1. Usando `lxml` "Sax Parser"

`lxml`[3] es una biblioteca bastante completa que permite analizar y trabajar con ficheros `XML` de manera cómoda. De hecho, implementa varios mecanismos distintos que permiten recorrer la estructura de un fichero `XML` así como tomar distintas acciones según el contenido del mismo.

Una de las maneras de manejar los ficheros XML es usando SAX [4, 5]. Este sistema de análisis de XML funciona de manera secuencial recorriendo las etiquetas. Usa una filosofía basada en *eventos* donde una función es llamada cada vez que aparece un nuevo elemento del documento XML.

Usando el siguiente código de ejemplo (basado en SAX), se pide que se realice un pequeño script en Python que lea de disco un feed RSS cualquiera y:

- Cuente el número de noticias o contenidos en el fichero RSS.
- Contabilice el número de imágenes que contiene.
- Busque algún término concreto (que se pueda pasar como parámetro al script) en los contenidos del RSS.
- Descargue las imágenes de los feeds a su disco duro (por ejemplo usando urllib[6]).

```
# etree sax parser
from lxml import etree

class ParseRssNews ():

        def __init__ (self):
                print ('---- Principio del archivo')

        def start (self, tag, attrib):      # Etiquetas de inicio
                print ('<%s>' % tag)
                for k in attrib:
                        print ('%s = "%s"' % (k,attrib[k]))

        def end (self, tag):                # Etiquetas de fin
                print ('</%s>' % tag)

        def data (self, data):              # texto
                print ('-%s-' % data)

        def close (self):
                print ('---- Fin del archivo')


parser = etree.XMLParser (target=ParseRssNews ())
etree.parse ('portada.xml', parser)

# 'portada.xml' es un rss en
# http://ep00.epimg.net/rss/elpais/portada.xml
```

2.2. Usando lxml "etree Parser"

Otra manera de manejar XML es a través de bibliotecas de manejo del DOM, donde el fichero XML se carga en memoria en una estructura de árbol que puede

ser recorrida fácilmente. La biblioteca `lxml` tiene la posibilidad de acceder a ficheros `XML` usando este tipo de acceso, usando el módulo `etree`[7].

Realice un script similar al del punto anterior pero usando esta tecnología `DOM`. El siguiente ejemplo puede ser útil:

```
from lxml import etree

tree = etree.parse('portada.xml')

# Root element
rss = tree.getroot()

# Los elementos funcionan como listas
# First child
channel = rss[0]

for e in channel:
       if (e.tag == 'item'):
              e.set('modificado', 'hoy')

              # Los atributos funcionan como diccionarios
              print (e.keys(), e.get('modificado'))

              otro = etree.Element('otro')

              otro.text = 'Texto de otro'
              e.insert(0, otro)


print (etree.tounicode(rss, pretty_print=True))
```

2.3. Usando Bibliotecas Especializadas

Existen varias bibliotecas especializadas que permiten acceder a contenidos `RSS` específicamente de manera muy sencilla [8]. Por ejemplo, la biblioteca **Universal Feed Parser** [9, 10] permite acceder a feeds `RSS` remotos de manera muy sencilla.

Implemente un script en Python que use dicha biblioteca y que realice las mismas acciones que en los puntos anteriores. Puede basarse en el código que existe en [8].

3. Integrando `RSS`s Externos en Nuestra Web

Basándonos en lo aprendido en los puntos anteriores, añadamos a nuestra web (de la práctica 3) algún cuadro en donde se muestren los últimos feeds de alguna fuente web que nos interese (periódico o web).

Sería conveniente, por no abusar del proveedor del `RSS`, no consultarlo con demasiada frecuencia (por ejemplo como máximo cada 10 minutos). Para ello podemos hacer algún tipo de caché en nuestra base de datos (o en disco) de los `RSS` que vayamos a mostrar.

Referencias

[1] RSS - Really Simple Syndication: Building and Using an RSS Feed: http://www.xul.fr/en-xml-rss.html

[2] Manual de RSS: http://www.desarrolloweb.com/manuales/manual-rss.html

[3] lxml - XML and HTML with Python: http://lxml.de

[4] SAX: http://www.saxproject.org/

[5] SAX: Simple API for XML: http://en.wikipedia.org/wiki/Simple_API_for_XML

[6] urllib: Open arbitrary resources by URL: http://docs.python.org/2/library/urllib.html

[7] The lxml.etree Tutorial: http://lxml.de/tutorial.html

[8] RSS Libraries: https://wiki.python.org/moin/RssLibraries

[9] Universal Feed Parser: http://code.google.com/p/feedparser/

[10] Universal Feed Parser Documentation: http://pythonhosted.org/feedparser/