Prácticas de XML
Servicios Telemáticos 2015-2016
Grado en Ingeniería Telemática
Universidad Rey Juan Carlos
`http://gsyc.urjc.es`

Observaciones

- Crea el fichero `~/st/practica02.txt`, que contendrá la memoria que escribas sobre este segundo bloque de prácticas. En la primera línea, indica tu nombre y login.

  En la memoria no es necesario que vuelvas a escribir el enunciado.

  Cuando la ejecución de un apartado de una práctica muestre algo en stdout, es necesario que copies esta salida y la pegues en la memoria.

  Crea el directorio `~/st/practica02` donde escribirás el resto de ficheros de esta práctica. Recuerda que la recogida de las prácticas se hará automáticamente el dia del examen, así que es muy importante que respetes al pie de la letra todos los nombres de ficheros y directorios que indiquen todos los enunciados.

Práctica 2.1. Holamundo xml

1. Escribe un fichero

   `~/st/practica02/coches.xml`

   que contenga los datos de un par de coches. Hazlo muy sencillo, prácticamente un *holamundo*, que contenga solo la matrícula, marca y modelo. Dale la estructura que consideres oportuna.

Práctica 2.2. xmlpp

Ahora usarás una herramienta de *pretty printing* para XML, esto es, un procesador que retoca un fichero para mejorar su presentación.

1. Si no tienes directorio `~/bin`, créatelo
2. Comprueba tu variable de entorno PATH ejecutando en la shell la orden `echo $PATH`

   Si `~/bin` no está incluido en tu PATH, añádelo. Para ello tendrás que editar el fichero `~/.bashrc` añadiendo la sentencia `export PATH=$PATH:$HOME/bin`
3. Descarga xmlpp[1]

   y guárdalo en `~/bin`
4. Añádele permiso de ejecución
5. Usando xmlpp, dale formato a

   `~/st/practica02/coches.xml`

   Para ello, debes invocar xmlpp, pasando el fichero a formatear como primer argumento.

   ```
   xmlpp ejemplo.xml
   ```

   Observa que xmlpp no modifica el fichero original, sino que muestra en stdout el fichero formateado. Usa redirecciones para que el fichero original quede formateado.

   Ten cuidado de no hacer nunca algo como esto

[1] `http://gsyc.es/~mortuno/docs/xmlpp`

```
xmlpp ejemplo.xml  > ejemplo.xml  # ¡¡ESTO ESTÁ MAL!!
```

Porque estarías leyendo y escribiendo simultáneamente el mismo fichero, lo que tendría resultados impredecibles. Tienes que llevar la salida a un fichero distinto, renombrándolo luego si es necesario.

Práctica 2.3. xmlcheck

Para comprobar que los documentos xml que escribes están bien formados, usaremos el script xmlcheck, que procesa el documento usando la librería cElementTree, muestra un mensaje si el fichero es correcto o levante una excepción en caso contrario

1. Descarga xmlcheck[2] y guárdalo en `~/bin`
2. Dale permiso de ejecución
3. Usando este script, comprueba que todos los documentos xml que escribas en esta práctica están bien formados (tanto los del apartado anterior como los de los apartados posteriores)

Práctica 2.4. discoteca.xml

1. Escribe el documento `~/st/practica02/discoteca01.xml`

   Contendrá los datos principales de un par de discos de música cualquiera. Googlea la información de los discos. Usa la estructura que te parezca adecuada, incluyendo algunos de los datos principales: titulo del disco, autor o autores, fecha de producción y/o publicación, canciones, duración, etc

2. Escribe el documento `~/st/practica02/discoteca02.xml`

   que contendrá la misma información, pero elige una estructura diferente. Puedes cambiar el uso de texto o atributos, los nombres de las etiquetas, etc. Obviamente, todos los documentos que escribas tienen que estar bien formados y tienes que tabularlos de forma claramente legible (con xmlpp) Además, todos los documentos tienen que respetar las indicaciones sobre XML que encontrarás en las transparencias (especialmente lo relativo a representar las *listas de cosas* como una serie de elementos del mismo tipo)

Práctica 2.5. mixmlpp

Hemos visto que el programa xmlpp es conveniente para dar formato a un documento xml. Pero no modifica el documento original, sino que envía el documento a la salida estándar.

Ahora vas a programar el script
`~/st/practica02/mixmlpp`
que usará xmlpp como módulo, para obtener nueva funcionalidad. Para que veas que en UNIX no es necesario que un script tenga extensión, usa exactamente ese nombre, mixmlpp, no xmlpp.py.

1. mixmlpp importará xmlpp, del que usará la siguiente función:

   ```
   pprint(xml, output=_sys.stdout, indent=4, width=80):
   ```

   El primer parámetro es una cadena de texto que contiene el documento xml completo.

   El segundo parámetro es un objeto fichero, ya abierto para escritura, donde se escribirá el fichero con formato. Si no se pasa este parámetro, la función usará la salida estándar.

   El tercer parámetro es el número de espacios correspondientes a un nuevo nivel de indentado. Si no se pasa este parámetro, la función usará 4.

   El cuarto parámetro es el tamaño del la lúneas. Si no se pasa, la función usará 80.

[2] `http://gsyc.es/~mortuno/docs/xmlcheck`

2. mixmlpp usará el módulo optparse o el módulo argparse.

   Como primer parámetro recibirá el fichero a leer.

   Si recibe un segundo parámetro, será el nombre del fichero donde escribirá la salida. Si no hay segundo parámetro, escribirá en stdout, como xmlpp.

   Con la opción `-m --modify`, dará formato al fichero original. Esto es, escribirá la salida en un fichero auxiliar y si no ha habido errores, sustituirá el fichero original por el auxiliar y borrará el auxiliar.

3. Si el script no puede modificar el fichero original, capturará la expcepción correspondiente y mostrará un mensaje por stderr.

4. Si el script recibe simultáneamete la opción `-m` y un segundo argumento, mostrará un error por stdout y morirá.

Práctica 2.6. mixmlpp desde stdin

Modifica mixmlpp para que si no recide ningún argumento, lea desde stdin y escriba en stdout

Práctica 2.7. procesa_coches

Escribe un script `~/st/practica02/procesa_coches.py`

Recibirá un nombre de fichero como primer argumento. Si no recibe ninguno, leerá desde stdin. Procesará ficheros con la estructura de tu `~/st/practica02/coches.xml` y mostrará toda la información por stdout, en forma legible y bien presentada para un humano, esto es, con estructura tabular y sin formato xml

Similar a esto:

```
Matricula  Marca   Modelo
-------------------------
9732CLM    Opel    Corsa
M0894MK    Seat    Panda
0371BCM    Honda   Civic
```

Práctica 2.8. procesa_discoteca

Escribe un script `~/st/practica02/procesa_discoteca.py`

Hará lo mismo que `procesa_coches.py`, pero con tu fichero `discoteca.xml`

La salida debería tener un aspecto más o menos parecido al siguiente, adaptado a la información que hayas decidido incluir en tu fichero discoteca.xml

```
Bill Evans /  The Bill Evans Album  / 1971 / 70:29
------------------------------------------------------------------
    1   Funkallero  7:49
    2   The Two Lonely People   6:13
    3   Sugar Plum  7:05
    [...]
    9   Re: Person I Knew (alternate)   7:21
    10  Funkallero (alternate)  6:1

Glenn Gould / J.S. Bach - The Goldberg Variations / 1981 / 51:03
------------------------------------------------------------------
    1   Aria    3:04
    2   Variatio 1 a 1 Clav.    1:10
    3   Variatio 2 a 1 Clav.    0:49
    [...]
    31  Variatio 30 a 1 Clav. Quodlibet 1:30
    32  Aria da capo
```

(El símbolo del corchete y los puntos suspensivos indica que en el ejemplo hemos omitido canciones, no lo tengas en cuenta)