UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Documentos XML: transformaciones con XSL/XSLT y Xalan

Apellidos, nombre	**Agustí i Melchor, Manuel** (magusti@disca.upv.es)
Departamento	**Departamento de Informática de Sistemas y Computadores**
Centro	**Escola Tècnica Superior d'Enginyeria Informàtica** Universitat Politècnica de València

1 Resumen

Un documento *eXtensible Markup Language o* XML[1] es uno cuya sintaxis está predefinida por el estándar, pero cuya estructura está definida por el usuario. Este puede crear un tipo de documento y hacerlo público, si quiere, en una *Document Type Declaration* (o DTD) para que otros puedan también escribir instancias de ese tipo de documento.

Sobre un **tipo de documentos XML** se puede desarrollar un sistema de intercambio de información entre dispositivos con diferentes capacidades o entre aplicaciones con diferentes necesidades y, como vamos a tratar en este caso, también **definir transformaciones** para ser aplicadas sobre los documentos de un tipo. Estas transformaciones pueden servir para convertir entre tipos de documentos "compatibles" (esto es que compartan algunas partes de su estructura) y, como es nuestro objetivo, en darle una apariencia que lo haga legible para el usuario final. Este es un caso habitual de los documentos para la web que originalmente se ofrecen como documentos HTML; pero que en función del dispositivo o la aplicación, puede ser necesario disponer también de una versión en PDF o en libro electrónico (*eBook/ePUB*).

Esto se puede hacer, sin utilizar XML, si se mantienen como documentos separados, lo que obliga a llevar cuenta de actualizar cada cambio en todas las versiones de los documentos. Una solución alternativa es partir de un único documento sin una representación (como es el caso de XML) y, sobre él, aplicar transformaciones. Eso lo hace especialmente indicado para situaciones en las que se espera poder disponer de varios formatos para el mismo contenido, como p. ej., a partir de XML se puede obtener una versión en texto ASCII, en otro vocabulario de XML (SVG, RSS, etc.), HTML, RTF, *eBook*, ODT, DOCX, TeX o PDF. Un documento escrito en lenguaje *eXtensible Stylesheet Language* (o XSL), será el que contenga las instrucciones para hacerlo.

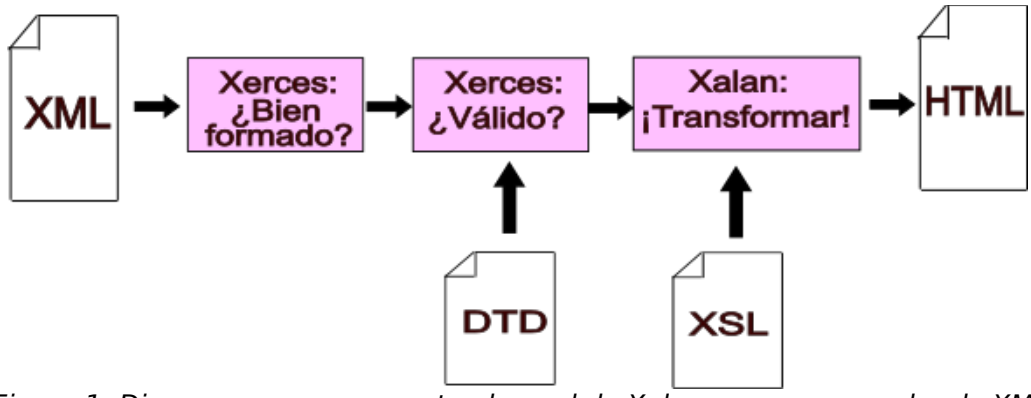

Figura 1: Diagrama que representa el papel de Xalan como procesador de XML y su relación con Xerces.

En este trabajo vamos a comentar cómo se puede gestionar esa transformación de documentos XML (como se muestra en la Figura 1 de forma esquemática) desde la línea de órdenes, utilizando *Xalan* como procesador de XML. Como paso previo se espera que se haya verificado la sintaxis del documento. Esta no es una tarea que realiza *Xalan*: se asume que una aplicación, como *Xerces*, lo habrá analizado previamente.

Aunque no abordaremos el detalle del uso de *Xerces* en este artículo, lo volveremos a mencionar cuando abordemos el uso práctico de *Xalan*, ya que este confiará a *Xerces* la tarea del análisis sintáctico.

---

1 Sobre su uso aplicado y los estándares derivados se pueden encontrar información en el sitio web de *xml.org* <http://www.xml.org/>.

Como requisitos para entender el contenido de este documento, se espera que el lector esté familiarizado con el uso lenguajes de marcado de documentos y los conceptos de bien formado y válido aplicables a un XML.

En el texto aparecen órdenes a ejecutar en un terminal, las reconoceremos por ir precedidas del signo '$'. Recordemos que el '$' no se ha de copiar al ejecutarlas en el terminal.

2 Objetivos

La estructura de un documento XML está claramente definida si utilizamos una DTD. Así que podemos diseñar transformaciones aplicables a un tipo de documento, puesto que se sabe qué partes debe o puede tener. Un mismo documento puede derivar en diferentes presentaciones finales, para lo cual se ha de diseñar la secuencia de instrucciones que extrae la información del XML y que guardaremos en un documento XSL. Una vez que el lector haya explorado los contenidos de este documento con atención, será capaz de:

- Instalar las herramientas informáticas que se sugieren para automatizar las tareas indicadas.
- Aplicar un subconjunto de las instrucciones de XSL para hacer posible la transformación mediante *Xalan*.

En lo que sigue, asumimos que estamos trabajando en una plataforma *GNU/Linux* donde ya existe una versión de *Java* instalada, pero la elección de las herramientas sugeridas permite estar trabajando en cualquier otra plataforma.

3 Introducción

En este punto, introduciremos la descripción de las herramientas necesarias para realizar las transformaciones sobre documentos XML que están bien formados y que son válidos para una DTD. Después introduciremos el lenguaje con el que se construyen las hojas de transformaciones de una forma práctica. Utilizaremos este mecanismo para transformar un documento XML y dotarle de un formato de presentación en HTML.

XML es [1] un lenguaje de marcado que describe una clase de objetos de datos llamados "documentos XML" y, parcialmente, el comportamiento de los programas que los procesan. XML es un metalenguaje que permite diseñar un lenguaje propio de etiquetas para múltiples clases de documentos. El estándar XML proviene de SGML[2] por lo que todo su contenido ha de estar escrito en forma de etiquetas con una serie de modificadores (llamados atributos). Las etiquetas pueden estar anidadas unas dentro de otras, pero toda etiqueta que se abra se tiene que cerrar y siempre en el mismo orden. Se puede encontrar en [2], [3] y [4] información ampliada sobre XML y su uso.

XML es el lenguaje que está facilitando el intercambio de información, permitiendo que exista una separación entre la capa de datos y la de presentación que permita ajustar esta segunda a las capacidades de cada dispositivo o aplicación que acceda a él. El uso de XML refuerza la separación entre contenido y presentación, al describir una estructura sin un formato predefinido y cerrado. Abordamos la transformación del documento XML a otros posibles formatos de documentos, asumiendo que el documento XML sigue una estructura descrita, p. ej., en una DTD [7].

---

2 Se puede leer sobre el estándar *Standard Generalized Markup Language (ISO 8879:1986)* en la Wikipedia <https://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language).

Para realizar esas transformaciones se podrían llevar a cabo en la máquina cliente desde un navegador o se podrían aplicar en el servidor utilizando un lenguaje como PHP, *Perl*, *Python* o *C/C++*. En este caso hemos optado por utilizar una aplicación independiente (un procesador de XML, en [5] encontraremos un amplio catálogo bastante actualizado) que pueda ser ejecutado desde la línea de órdenes como *Xalan*[3]. Hemos elegido este porque puede ser utilizado en las diferentes plataformas de escritorio y es parte del proyecto *Apache*[4], por lo que se le puede encontrar detrás de muchos servidores web actuales en funcionamiento.

Para ilustrar los ejemplos vamos a suponer que se va a trabajar sobre una posible DTD que denominaremos "trabajo de asignatura" y cuyo contenido se muestra (a dos columnas) en la Figura 2. Esta podría servir para gestionar los trabajos que en una asignatura se estuvieran llevando a cabo en un determinado curso académico. En él, toda la información parte del título del trabajo que, a su vez, se define en base a las personas que lo realizan (los componentes) y las palabras clave que lo describen. La información necesaria para entenderlo se agrupa en la introducción. Como es un trabajo práctico, contiene los requisitos para ponerlo en marcha o ejecutarlo y la descripción de los archivos que componen el trabajo. Algunas de estas características del trabajo se deberán utilizar obligatoriamente, otras no. Y, algunas partes pueden repetirse porque, como los componentes o las palabras clave, pueden utilizarse en un número, en principio, indeterminado. Que conste que es una decisión arbitraria, no lo tomemos como una limitación.

```
<!ELEMENT Trabajo (Titulo, Grupo,
      PalabrasClave?,Introduccion,
      Desarrollo?, Archivos?)>
<!ELEMENT Titulo (#PCDATA)>
<!ELEMENT Grupo (Componente+)>
<!ELEMENT  Componente  (Nombre,  Datos?,
                      foto?)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Datos EMPTY>
<!ATTLIST Datos
        Asignatura CDATA #REQUIRED
        Titulacion (Inf | Doc) #IMPLIED
        Correo CDATA #IMPLIED >
```

```
<!ELEMENT foto EMPTY>
<!ATTLIST foto
          fichero  CDATA #REQUIRED
          ancho    CDATA #REQUIRED
          alto     CDATA #REQUIRED >
<!ELEMENT PalabrasClave (#PCDATA)>
<!ELEMENT  Introduccion  (Resumen,
                      Requerimientos? )>
<!ELEMENT Resumen (Parrafo+)>
<!ELEMENT Requerimientos (Parrafo+)>
<!ELEMENT Desarrollo (#PCDATA)>
<!ELEMENT Archivos (#PCDATA)>
```

Figura 2: DTD "trabajo de asignatura" que se utilizará en este trabajo.

3.1 Herramientas para procesar documentos XML

Para maximizar las opciones de uso de las acciones que aquí se proponen, la forma de realizar las acciones será en la línea de órdenes, por lo que necesitaremos un terminal. También necesitamos un editor y un procesador de XML.

Cualquier editor de texto nos sirve, pero existen aplicaciones especializadas para ayudarnos a escribir y minimizar los errores propios de la edición manual. Así, p. ej., *emacs*, *kate* y la mayoría de editores de código fuente para desarrollar aplicaciones se pueden utilizar con buenos resultados. Con cualquiera de ellos se puede copiar el contenido del Listado 1 en un documento "trabajoV2.xml". Los documentos XML se sugiere que empiecen con una línea

3 Pero que conste que hay otras opciones como, p. ej., *Saxon*, *xt*, *sablotron*, etc. de los que cualquier buscador te dará más información.

4 Para más información al respecto, véase el sitio web del proyecto en <http://www.apache.org/>.

que describe la versión de XML y el tipo de codificación del documento. Es una instrucción especial y la reconoceremos por su nombre "?xml". Aquí utilizaremos el conjunto de caracteres ISO-8859-1 porque es el que incluye los caracteres propios del alfabeto español; facilitando así, entre otras cosas, la utilización de caracteres acentuados. En este caso, el elemento raíz está representado por la etiqueta *Trabajo*. El resto de elementos representan información relativa al título, a los componentes del grupo y a los párrafos del resumen El elemento *Titulo* contiene una cadena de caracteres Los elementos *Grupo* y *Resumen* constan de otro nivel de descendientes (*Componente* y *Parrafo*, respectivamente), que a su vez contienen cadenas de caracteres.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Trabajo>
      <Titulo> Un ejemplo básico de documento escrito en XML </Titulo>
      <Grupo>
            <Componente> <Nombre>Steve Rogres</Nombre>  </Componente>
            <Componente> <Nombre> Anthony Strak</Nombre> </Componente>
            <Componente> <Nombre>Yo Mismo También</Nombre> </Componente>
      </Grupo>
      <Introduccion>
            <Resumen>
            <Parrafo>El trabajo consiste en explicar cómo procesar un
documento XML.</Parrafo>
            <Parrafo>Para ello, se scribirá un documentos XML, como
este.</Parrafo>
            <Parrafo>A continuación se transformará con una XSL y
Xalan.</Parrafo>
            </Resumen>
      </Introduccion>
</Trabajo>
```

Listado 1: Contenido del documento trabajoV2.xml.

Para aplicar las transformaciones, en el caso de optar por *Xalan* [6], se puede optar por la versión *Xalan-J* que se ejecuta sobre *Java* por que sea lo más portable posible todo lo que se comenta en este documento. Si no tuvieras Java instalado, en la versión Ubuntu 18.04 de Linux es posible instalar varias alternativas, como por ejemplo:

```
$ apt install openjdk-11-jre
```

Para instalar *Xalan-J*, se puede descargar el binario de nombre *Xalan-J_2_0_0*[5] y descomprimirlo, p. ej. como muestra el Listado 2, cuya última instrucción es para comprobar la instalación: si obtienes la ayuda que se muestra, todo está bien instalado. Como *Xalan* tiene un largo formato para la orden, utilizaremos un *script*[6] (véase en el Listado 3 el contenido del mismo) llamado *xalan.sh*.

Así que en un terminal podremos lanzar la orden que llevará a cabo la transformación sobre el contenido de un fichero (p. ej., de nombre *fitxer.xml*) como:

```
$ xalan.sh fitxer.xml fitxerTransf.xsl documentoDeSalida.html
```

5 La podemos encontrar en *The Apache Xalan Project* y, más concretamente, en <http://archive.apache.org/dist/xml/xalan-j/xalan-j_2_0_0.tar.gz>.

6 Si vamos a utilizar *macOS*, podemos seguir los mismos pasos que aquí mostramos. Si vamos a utilizar la plataforma *MS/Windows*, pero solo en caso necesario ;-), a través del correo electrónico, podríamos enviar la versión de *xalan.bat* para esa plataforma. El resto también nos sirve.

donde utilizaremos un fichero *fitxerTransf.xsl* que contendrá la transformación a aplicar y el resultado del proceso se guardará en el fichero *documentoDeSalida.html*. De modo que un mismo fichero XML podrá ser transformado por diferentes ficheros XSL y, a cada uno, le podremos asignar un nombre de fichero de resultado diferente. En este caso es HTML (y sin florituras, porque es lo más breve de exponer en este documento), pero podría ser cualquier otro de los formatos enumerados al principio.

```
$ mkdir libs
$ wget http://archive.apache.org/dist/xml/xalan-j/xalan-j_2_0_0.tar.gz
$ tar xvf  xalan-j_2_0_0.tar.gz -C libs
$ java -cp libs/xalan-j_2_0_0/bin/xalan.jar org.apache.xalan.xslt.Process
=xslproc options:
    -IN inputXMLURL
  [[  He suprimido parte de la salida por brevedad de la exposición]]
```

Listado 2: Instalación y comprobación de Xalan-J.

```
#!/bin/sh
JAVA=java
JARS_HOME= libs/xalan-j_2_0_0/bin/xalan.jar:\
libs/xalan-j_2_0_0/bin/xerces.jar: libs/xerces-1_4_4/xercesSamples.jar
 if [ $# -lt 1 ];
  then
   echo "Procesar un documento XML mediante Xalan:"
   $JAVA -cp $JARS_HOME org.apache.xalan.xslt.Process
   exit -1
 fi
$JAVA $JARS_HOME org.apache.xalan.xslt.Process $*
```

Listado 3: Contenido del script xalan.sh.

3.2 Un lenguaje de programación para las transformaciones sobre XML: XSL/XSLT

XSL es un lenguaje o vocabulario de XML para transformar documentos XML. Es decir, se utiliza para transformar o reexpresar el contenido de un documento en un formato conocido. Para ello, a diferencia de las hojas de estilo (*Cascading Style Sheet* o CSS), las hojas de transformaciones (ficheros con extensión *.xsl*), permiten crear una salida que sea el resultado de seleccionar sobre qué parte del documento de partida se trabaja, componerlo con parte o la totalidad de otros, ... y describir su formato (su presentación) si es la intención del autor.

XSL está compuesto de tres partes. La más visible es XSLT (*XSL Transformations*) y consiste en la definición de un vocabulario de XML que permite indicar a *Xalan* la transformación a aplicar sobre el documento (o los documentos) de partida; esto es, las reglas que van a implementar la transformación a aplicar. Además, está *XPath (XML Path Language) que define* las expresiones que permiten indicar los puntos a los que se refieren las acciones. Y, en tercer lugar, XSL-FO *(XSL Formatting Objects*), que es un vocabulario especializado en la descripción de los elementos de presentación que se utilizan en la edición sobre formato papel.

Las transformaciones que se describen con XSL son una secuencia de instrucciones, como el código fuente de un programa, pero bajo la sintaxis de XML. Se basan en la utilización de tres elementos constructivos:

- Las variables y parámetros del proceso, como en otros lenguajes de programación.
- Los selectores o expresiones que permiten obtener el contenido de un nodo, el de uno de sus atributos o especificar un conjunto de los primeros.
- Los patrones o reglas de selección de los elementos del árbol XML que se está procesando,

Para facilitar la diferenciación entre código XSL, propiamente dicho, del resto del documento, en las XSL que utilizaremos se antepondrá siempre el prefijo **xsl:** a las etiquetas de este vocabulario. Esto se conoce como especificación del espacio de nombres y se utiliza para evitar conflictos con etiquetas que pudieran denominarse igual pero que sean parte del resultado de la transformación.

4 Transformaciones sobre un documento XML

El documento se puede considerar que está construido acorde con un vocabulario o instanciación de un tipo de documento si se ha comprobado que sigue las reglas de sintaxis XML y que cumple con las de definición de una gramática. Esto es, respectivamente, si está "bien formado" (*well-formed*) y es valido. Ambas comprobaciones las puede realizar un analizador (o *parser*) como *Xerces*.

Con el fin de obtener diferentes representaciones de la información en formatos (como HTML, PDF, etc.) utilizaremos "scripts" XSL [8], es decir, programas que utilizan como datos de entrada documentos XML y obtienen como salida el formato elegido. Vamos a ver, ahora, algunos ejemplos de uso de código XSL/XSLT para convertir de XML a TXT y a HTML. Para ejecutarlos utilizaremos a Xalan.

4.1 Una transformación XML a TXT

Los documentos XSL están compuestos de reglas o plantillas denominadas *templates*, que indican cómo se procesa (transforma) un nodo de un documento XML. Estas plantillas describen la transformación a aplicar de manera detallada. Una transformación XSL puede añadir nuevos elementos como resultado de su aplicación sobre un documento XML de partida o eliminar elementos, puede también reordenar los elementos, realizar comprobaciones y tomar decisiones acerca de qué elementos mostrar, etc. Por ejemplo, podemos obtener una versión en texto ASCII de nuestro documento de partida con una hoja de transformaciones como la que muestra el Listado 4.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="text" indent="yes" encoding="ISO-8859-1"/>
 <xsl:template match="/">
  ¡Hola, mundo!
áéíóú àèìòù
 </xsl:template>
</xsl:stylesheet>
```

Listado 4: Contenido de la hoja de transformaciones xsl1.xsl.

Observe que es un fichero XML, que está organizado alrededor de un elemento raíz **stylesheet**. En un segundo nivel, hay dos elementos: **output** y **template**. El primero sirve para indicar las características propias del documento que se va a generar como resultado de aplicar esta transformación. El segundo es el punto de entrada o de inicio de todo el código que realiza la transformación.

Este segundo elemento tiene un atributo **match** que indica sobre qué elemento del documento XML original se aplica. En este caso la especificación del elemento raíz del documento ("/") representa a la totalidad del documento.

Si el fichero que guarda esta transformación se llama *xsl1.xsl*, se obtiene una transformación que, independientemente del contenido del documento de partida, obtiene el resultado que muestra el Listado 5.

```
$ xalan.sh trabajoV2.xml xsl1.xsl resultado1.txt
¡Hola, mundo!
áéíóú àèìòù
```

Listado 5: Resultado de aplicar la hoja de transformaciones xsl1.xsl..

4.2 Una primera transformación a HTML

Para que la salida obtenida dependa del contenido del documento en cuestión, debe ir accediendo al contenido de los elementos del documento XML, p. ej., como la que muestra el Listado 6 y que aplicaremos sobre el documento con una orden parecida a la que se ha utilizado en el Listado 5.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" indent="yes" encoding="ISO-8859-1"
                    doctype-public="-//W3C//DTD HTML 4.01//EN"
                    doctype-system="”http://www.w3.org/TR/html4/strict.dtd"  />
 <xsl:template match="/">
   <html>
    <head>
      <title><xsl:value-of select="Trabajo/Titulo"/></title>
    </head>
    <body>
     <h1><xsl:value-of select="Trabajo/Titulo"/></h1>
     <p><xsl:value-of select="Trabajo/Introduccion/Resumen/Parrafo”/></p>
    </body>
   </html>
 </xsl:template>
</xsl:stylesheet>
```

Listado 6: Contenido de la hoja de transformaciones xsl2.xsl.

Como en el caso anterior, en el contenido del fichero XSL (Listado 6), se propone la identificación del tipo de documento a generar (**output**) y una secuencia de instrucciones de transformación dentro de una única plantilla (**template**). A diferencia del caso del apartado anterior, lo que se genera es código para construir un documento HTML, así que hay que ir generando las etiquetas propias de este vocabulario de SGML e ir asignando el contenido de estas etiquetas con las selecciones del documento XML que se realizan con

value-of. Observemos cómo se utilizan las especificaciones de qué elementos están involucrados con las expresiones *XPath*: como si fuesen rutas del sistema de archivos de un sistema operativo, se especifican de forma absoluta o relativa al punto del árbol XML al que se aplica una plantilla.

4.3 Una segunda transformación a HTML

Para que el proceso se extienda a otros elementos del documento hay que hacer lo propio con estas secuencias de definición de plantillas (reglas o patrones) junto a las instrucciones que engloban.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" indent="yes" encoding="ISO-8859-1"
                    doctype-public="-//W3C//DTD HTML 4.01//EN"
                    doctype-system=”http://www.w3.org/TR/html4/strict.dtd" />
 <xsl:template match="/">
   <html>
    <head>
      <title><xsl:value-of select="Titulo"/></title>
    </head>
    <body>
     <!-- Para  indicar que  en este punto se han de lanzar  otras plantillas  -->
     <xsl:apply-templates />
    </body>
   </html>
 </xsl:template>
 <xsl:template match="Titulo">
  <h1><xsl:value-of select="."/></h1>
 </xsl:template>
<xsl:template match="Grupo | Componente">
      <xsl:apply-templates />
 </xsl:template>
<xsl:template match="Nombre">
  <p class="nom">
    Jo soc:<xsl:value-of select="."/>.
  </p>
 </xsl:template>
 <xsl:template match="Datos">
  <xsl:if test="@Correo">
     <p class="datos">  Escríbeme al
      <a href="mailto:{@Correo}">  <xsl:value-of select="@Correo"/> </a>.
     </p>
  </xsl:if >
 </xsl:template>
 <xsl:template match="Foto | Introduccion">
 </xsl:template>
</xsl:stylesheet>
```

Listado 7: Contenido de la hoja de transformaciones xsl3.xsl.

Veamos una nueva transformación en el Listado 7. En él vemos:

- Que para indicar que hemos de lanzar otras plantillas (*templates*), se utiliza la instrucción *<xsl:apply-templates />*. Que *las* plantillas pueden ser simples, como la que aplicaremos al aparecer un elemento de etiqueta *Titulo*, o compuestas, como la que se lanza al aparecer tanto la etiqueta *Grupo* como *Componente.* Y que pueden no generar nada,

como sucede con la plantilla que se activa con la etiqueta *Foto* o *Introducción* indistintamente.

- Seleccionar el contenido del elemento actual del documento de partida con *<xsl:value-of select="."/>*.
- Especificar una acción condicionada a la evaluación de una expresión con *<xsl:if test= [[valor]] >* y ver cómo acceder *a* al contenido de un atributo con la '@'.

5 Conclusión

A lo largo de este artículo se ha visto cómo se puede procesar o transformar un documento XML. De esta forma, se mantiene la separación entre la capa de datos y la de presentación. Por brevedad en la exposición solo se han generado transformaciones a TXT y HTML, pero se puede hacer a cualquier otro formato de fichero que se conozca cómo se ha de escribir.

En este trabajo se ha comentado cómo se puede gestionar esa transformación de documentos XML desde la línea de órdenes utilizando *Xalan*. De esta forma, modificando el contenido se pueden lanzar *scripts* que regeneren todas las versiones finales, cada una con sus propias características de estilo y formato. Ahora hemos de dejarlo, pero sugiero aventurarse por cuenta propia, viendo cómo ejecutar las transformaciones para comprobar qué sale y explorar el uso del lenguaje XSL practicando.

6 Bibliografía

[1] W3C. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C Recommendation*. Disponible en <https://www.w3.org/TR/REC-xml/>.

[2] J. Merelo. (2000). Transformando documentos XML usando XSLT. Grupo GeNeura. Disponible en <http://geneura.ugr.es/~jmerelo/XSLT/XSLT-2001-1ed.htm>.

[3] J. Merelo. (2000). Generación de páginas Web usando XSLT y XML. Grupo GeNeura. Disponible en <http://geneurael.ugr.es/~jmerelo/XSLT/ >.

[4] E. R. Harold. (2004). Café con Leche XML. Disponible en <http://www.cafeconleche.org/>.

[5] L. M. Garshol.(2005). *XML tools by category*. Disponible en <http://www.garshol.priv.no/download/xmltools/cat_ix.html >.

[6] *Apache Xalan Project*. Disponible en <http://xalan.apache.org/index.html>.

[7] M. Nic (traducción de I. García). *DTD Tutorial*, Disponible en <http://www.zvon.org/xxl/DTDTutorial/General_spa/book.html>.

[8] P. Grosso y N. Walsh. (2000). *XSL Concepts and Practical Use. Arbortext. Disponible en* <https://nwalsh.com/docs/tutorials/xsl/xsl/slides.html>.