5. Conversión y adaptación de documentos XML

Autor	Ⓧ Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Feb 23, 2021 7:02 AM

1. Introducción

Si queremos usar directamente los datos de un documento XML es necesario transformarlo primero.

Los navegadores interpretan las etiquetas del documento XML aplicando un formato especificado en las hojas CSS. Es posible transformar un documento

XMl en otro tipo de documento. Algunas de las tecnologías que lo permiten son:

- **XSLT:** permite definir el modo de transformar un documento XML en otro.
- **XSL-FO:** permite transformar un documento XML en otro documento de formato legible e imprimible, por ejemplo PDF.
- **Xpath:** permite acceder a diversos componentes de un XML.

A día de hoy se usa XSLT, estándar aprobado por W3C. y dichos documentos se llaman hojas XSLT.

XSLT deriva de XML, por tanto también son documentos XML, al igual que los documentos XSD, RSS o atom.

A partir de usar XSLT podemos generar otro documento XML, HTML o un documento de texto.

2. Estructura básica de una hoja XSLT.

- Elementos XSLT. Precedidos del prefijo xsl: pertenecen al espacio de nombres xsl, definidos en el estándar del lenguaje e interpretados por cualquier procesador XSLT.
- Elementos LRE, no pertenecen a XSLT, sino que se repiten en la salida sin más.
- Elementos de extensión, no pertenecen al espacio de nombres xsl, son manejados por implementaciones concretas del procesador. Normalmente no se usan.

Para asociar un XML a una hoja XSLT se usa la siguiente línea:

```
<?xml-stylesheet type="text/xsl" href="ruta_del_fichero_xsl.xsl"?>
```

3. Elementos XSLT

El elemento raíz es xsl:stylesheet o xls:transform. Sus atributos principales son:

- **version,** puede ser 1.0 o 2.0.

- **xmlns:xsl**, se utiliza para declarar el espacio de nombres xsl. Para XSLT suele ser la dirección: http://www.w3.org/1999/XSL/Transform

A los elementos hijos se les conoce como elementos de nivel superior, estructuras contenedoras de instrucciones. Si son hijos directos no se pueden anidar, excepto xsl: variable y xls:param.

- **xsl:attribute,** añade un atributo a un elemento en el árbol de resultados.
- **xsl:choose,** permite decidir que parte de la hoja XSL se va a procesar en función de los resultados
- **xsl:decimal-format,** define un patrón que permite converit en cadenas de texto números en coma flotante.
- **xsl:for-each**, se aplica sentencias a cada uno de los nodos del árbol que recibe como argumento.
- **xsl:if,** permite decidir si se va a procesar o no una parte del XSl en función de una condición.
- **xsl:import,** importa una hoja de estilos XSLT utilizada en una URI dada.
- **xsl:key,** define una o varias claves para ser referenciadas desde cualquier lugar del documento.
- **xsl:output**, define el tipo de salida que se genera como resultado.
- **xsl:preserve-space**, especifica cuales son los elementos del documento XML que no tienen espacios en blanco eliminados antes de la transformación.
- **xsl:strip-space,** especifica cuales son los elementos del documento XML que tienen espacios en blanco eliminados antes de la transformación.
- **xsl:template,** es el bloque fundamental de una hoja XSLT.
- **xls:value-of,** calcula el valor de una expresión Xpath dada y lo inserta en el árbol de resultados del documento de salida.
- **xsl:variable,** asigna un valor a una etiqueta para usarlo más cómodamente.

4. XPath

Es un estándar, diferente de XML aprobado por el W3C, que permite navegar entre los elementos y atributos de un XML.

Para hacerlo se basa en relaciones de parentesco entre los nodos del documento.

En la actualidad se utiliza con XLST, XML Schema, Xquery, Xlink, Xpointer, Xforms, etc.

Expresiones de camino

Xpath se usa definiendo expresiones de camino para seleccionar nodos o conjuntos de no de un documento XML. las cuales se parecen mucho a las path de los sistemas de fichero.

Estas expresiones se aplican a un XML, asumiendo que su estructura interna es en árbol. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<matriculas>
  <alumno>
    <nombre>Pedro</nombre>
    <apellido1>López</apellido1>
    <apellido2>Ortega</apellido2>
    <DNI pais="es">López</DNI>
  </alumno>
</matriculas>
```

Al evaluar la expresión /matriculas/alumno/nombre podemos obtener los nombres de alumnos matriculados.

4.1. Términos básicos

Nodos

Un documento XML es tratado en Xpath como un árbol de nodos. Hay 7 tipos: elemento, atributo, texto, espacio de nombres, instrucción de proceso, comentario y documento. El elemento más alto en el árbol es el nodo raíz.

- **Nodo raíz.** Contiene el ejemplar del fichero XML, no hay que confundir con el elemento raíz del documento, el cual está debajo de él.
- **Nodo elemento,** cada uno de los elementos del documento XML.
  - Tienen un elemento padre
  - El padre del elemento raíz es el nodo raíz del documento.
  - Pueden tener identificadores únicos para referenciarlos de forma más directa. Es necesario que entonces el atributo esté definido en un DTD

o un XSD asociado.

- **Nodos texto.** Aquellos caracteres que no están marcados con ninguna etiqueta. No tienen hijos.
- **Nodos atributos,** son los atributos de un elemento.
  - Se consideran etiquetas añadidas al nodo elemento.
  - No se consideran hijos de ese elemento
  - Aquellos atributos con valor asignado en el esquema asociado se tratarán como si ese valor se le hubiese dado al escribir el XML.
  - Para las definiciones de espacios de nombre y para aquellos atributos definidos con la propiedad #IMPLIED en su DTD no se crean nodos.

```
<?xml version="1.0" encoding="UTF-8"?>
<matriculas>
  <alumno>
    <nombre>Pedro</nombre>
    <apellido1>López</apellido1>
    <apellido2>Ortega</apellido2>
    <DNI pais="es">López</DNI>
  </alumno>
</matriculas>
```

<matrículas> es el nodo elemento raíz.

<nombre>Pedro</nombre> es un nodo elemento.

pais="es" es un nodo atributo.

Ítems

Los ítems pueden ser nodos o valores atómicos

En el caso ejemplo, valores atómicos son "Pedro" o "es"

Relaciones entre nodos

Según como se sitúen los nodos dentro del árbol se habla de relación entre ellos:

- <nombre> es hijo de <alumno>
- <nombre> y <DNI> son hermanos
- <matricula> es un ascendente de <nombre>
- <apellido1> es descendiente de <matriculas>

4.2. Expresiones

Los resultados que da la evaluación de una expresión son:

- Un conjunto de nodos (node-set)
  - No está ordenado.
  - Se considera que todos los elementos de un conjunto de nodos son hermanos, independientemente de lo que fueran originalmente.
  - Aunque los hijos de los nodos formar un conjunto de nodos son accesibles, los subárboles de un nodo no se consideran elementos del conjunto.
  - Un valor booleano.
  - Un número.
  - Una cadena.

  Elementos a utilizar en una expresión Xpath:

  - Agrupaciones: (), {}, [].
  - Elemento actual, elemento padre.
  - Atributos: @.
  - Elementos "*"
  - Separadores "::"
  - Comas ","
  - El nombre de un elemento.
  - Tipo de nodo, que puede ser:
    - comment
    - text
    - procesing instruction
    - node
  - Operadores: and, or, mod, div, *, /, //, |, +, -, =, !=, <, >, <=, >=
  - Nombres de función.
  - Denominación de ejes: ancestor, ancestor-or-self-atribute, child, descendant, descendant-or-self, following, following-sibling,

namespace, parent, preceding, preceding-sibling, self.

- Literales: se ponen entre comillas dobles o simples. Pueden anidare alternando el tipo de comillas.
- Números.
- Referencias a variables, se utiliza; $nombreVariable.

4.3. Procesando expresiones

- Nodo actual, es aquél en el que se encuentra el procesador.
- Nodo contexto, cada expresión está formada por subexpresiones que se van evaluando antes de resolver la siguiente.
- Tamaño del contexto, es el número de nodos qeu ese están evaluando en un momento dado en una expresión Xpath.

4.4. Ruta de localización

La ruta de localización de un nodo es la ruta que hay que seguir en un árbol para localizarlo. Se evalúan y esa evaluación devuelve un conjunto de nodos, o vacío.

Una ruta de localización se realiza mediante varios pasos de localización:

- **Ruta de localización del nodo raíz del documento:** Es la barra diagonal del documento, se trata de una ruta absoluta.
- **Localización de un elemento,** selecciona todos los hijos del nodo de contexto con el nombre especificado. los elementos a los que se refiera dependerán de la localización del nodo de contexto (ruta relativa). Puede devolver un elemento vacío.
- **Localización de atributos,** para referirnos a ellos se utiliza el símbolo @ seguido de su nombre.
- **Localización de espacios de nombres,** no se tratan explícitamente.
- **Localización de comentarios**
- **Localización de nodos de texto.**
- **Localización de instrucciones de procesamiento.**

Para comparar distintos elementos y tipos de nodo simultáneamente se utilizan los siguientes comodines:

- **Asterisco**, compara cualquier nodo de elemento, independientemente de su nombre. No compara atributos, comentarios, nodos de texto o instrucciones de procesamiento.
- **Node():** compara, además de los tipos de elementos, el nodo raíz, los nodos de texto, los de instrucción de procesamiento, nodos de espacio de nombre, los atributos y comentarios.
- **@** compara los nodos de atributo.

4.5. Funciones del XPath

Podemos emplearlas unidas a las rutas de localiación para hacer cosas sobre conjuntos de elementos que devuelven predicados.

Las más importantes son:

- **boolean(),** al aplicarla sobre un conjunto devuelve true si no es vacío.
- **not(),** al aplicarla sobre un predicado devuelve true si es falso y falso si el predicado es true.
- **true(),** devuelve el valor true
- **false(),** devuelve el valor false
- **count()**, devuelve el número de nodos que forman un conjunto de nodos.
- **name(),** devuelve un nombre de un nodo.
- **local-name(),** devuelve el nombre del nodo actual o del primer nodo de un conjunto de nodos.
- **namespace-uri(),** devuelve el URI del nodo actual o del primer nodo de un conjunto dado.
- **position(),** devuelve la posición de un nodo en su contexto comenzando en 1. Por ejemplo, para seleccionar los dos primeros elementos de tipo elemento de un fichero XML pondremos: //elemento[position()<=2]
- **last(),** Devuelve el último elemento del conjunto dado.
- **normalize-space(),** permite normalizar los espacios de una cadena de texto, es decir, si se introduce una cadena donde hay varios espacios consecutivos, esta función lo sustituye por uno solo.
- **string()**, es una función que convierte un objeto en una cadena. Los valores numéricos se convierten en la cadena que los representa teniendo

en cuenta que los positivos pierden el signo. Los valores booleanos se convierten en la cadena que representa su valor, esto es “true” o “false”

- **concat()**, devuelve dos cadenas de texto concatenadas. El ejemplo siguiente devuelve “XPath permite obtener datos de un fichero XML”.
- **string-length(),** devuelve la cantidad de caracteres que forman una cadena de caracteres.
- **sum()**, devuelve la suma de los valores numéricos de cada nodo en un conjunto de nodos determinado.

4.6. Predicados

Es una expresión booleana que añade un nivel de verificación al paso de localización, en las que podemos incorporar funciones XPath.

La ventaja que ofrece es permitir seleccionar un nodo que cumple ciertas características.

Los predicados se incluyen dentro de una ruta de localización utilizando corchetes. Ejemplo:

/receta/ingredientes/ingrediente[@codigo=“1”]/nombre

En un predicado se pueden incluir:

- **Ejes**. Permiten seleccionar el subárbol dentro del nodo contexto que cumple un patrón. Puede ser o no de contenido:
  - **child,** es el eje por defecto, su forma habitual es la barra, aunque también puede ponerse /child::
  - **attribute,** permite seleccionar los atributos que deseemos, es el único eje que no es de contenido.
  - **descendant**, permite seleccionar todos los nodos que descienden del conjunto de nodos contextos. Se corresponde con la doble barra, “//”, aunque se puede usar: descendant::
  - **self,** se refiere al nodo contexto y se corresponde con el punto ”.”.
  - **parent**, selecciona los nodos padre, para referirnos a él usamos los dos puntos, ”..”
  - **ancestor,** devuelve todos los nodos de los que el nodo contexto es descendiente.

- **Nodos test,** permiten restringir lo que devuelve una expresión XPath, se pueden agrupar en función de los ejes a los que se puede aplicar.

  Aplicable a cualquier eje:

  - ***** solo devuelve elementos, atributos o espacios de nombres.
  - **nod()** devuelve todos los nodos de todo tipo.

  Aplicable a ejes de contenido:

  - **text(),** devuelve cualquier nodo de tipo texto.
  - **comment()** devuelve cualquier nodo de tipo contenido.
  - **processing-instruction()** devuelve cualquier tipo de instrucción de proceso.

Varios predicados se pueden unir mediante operadores lógicos: and, or o not.

4.7. Ejercicio resuelto

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!DOCTYPE agenda>
<agenda>
  <propietario>
    <identificadores>
      <nombre>Alma</nombre>
      <apellidos>López Terán</apellidos>
    </identificadores>
    <direccion>
      <calle>El Percebe 13, 6F</calle>
      <localidad>Torrelavega</localidad>
      <cp>39300</cp>
    </direccion>
    <telefonos>
      <movil>970898765</movil>
      <casa>942124567</casa>
      <trabajo>628983456</trabajo>
    </telefonos>
  </propietario>
  <contactos>
    <persona id="p01">
      <identificadores>
        <nombre>Inés</nombre>
        <apellidos>López Pérez</apellidos>
      </identificadores>
      <direccion>
        <calle>El Ranchito 24, 6B</calle>
        <localidad>Santander</localidad>
        <cp>39006</cp>
```

```
      </direccion>
      <telefonos>
        <movil>970123123</movil>
      </telefonos>
    </persona>
    <persona id="p02">
      <identificadores>
        <nombre>Roberto</nombre>
        <apellidos>Gutiérrez Gómez</apellidos>
      </identificadores>
      <direccion>
        <calle>El Marranito 4, 2F</calle>
        <localidad>Santander</localidad>
        <cp>39004</cp>
      </direccion>
      <telefonos>
        <movil>970987456</movil>
        <casa>942333323</casa>
      </telefonos>
    </persona>
    <persona id="p03">
      <identificadores>
        <nombre>Juan</nombre>
        <apellidos>Sánchez Martínez</apellidos>
      </identificadores>
      <direccion>
        <calle>El Cangrejo 10, sn</calle>
        <localidad>Torrelavega</localidad>
        <cp>39300</cp>
      </direccion>
      <telefonos>
        <movil>997564343</movil>
        <casa>942987974</casa>
        <trabajo>677899234</trabajo>
      </telefonos>
    </persona>
  </contactos>
</agenda>
```

Construir las sentencias **XPath** que permitan obtener los siguientes datos:

1. Nombre del propietario de la agenda.
2. Teléfono de casa del propietario.
3. Nombres y apellidos de los contactos de la agenda.
4. Nombre e identificador de cada contacto.
5. Datos del contacto con identificador "p02".
6. Identificadores de los contactos que tienen móvil.

- Nombre del propietario de la agenda.

```
/agenda/propietario/identificadores/nombre
```

- Teléfono de casa del propietario.

```
/agenda/propietario/telefonos/casa
```

- Nombres y apellidos de los contactos de la agenda.

```
//contactos/persona/identificadores/nombre |
//contactos/persona/identificadores/apellidos
```

- Nombre e identificador de cada contacto.

```
//contactos/persona/identificadores/nombre | //contactos/persona/@id
```

- Datos del contacto con identificador "p02".

```
//contactos/persona[@id="p02"]/*/*
```

- Identificadores de los contactos que tienen teléfono en casa.

```
//contactos/persona/telefonos/casa/../../@id
```

4.8. Acceso a datos desde otro documento XML

Es útil poder acceder a los datos desde otros ficheros. Para ello se usa la función document(), pero no es del lenguaje XPath, sino que pertenece a XSLT.

Esta función admite dos argumentos:

document(URI) - devuelve el elemento raíz del documento XML que se localiza en el URI especificado.

document(nodo) - devuelve el conjunto de nodos cuya raíz es el nodo dado.

5. Utilización de plantillas

El elemento xsl:template controla el formato de salida que se le aplica a ciertos datos de entrada. Para especificar la salida se utilizan sentencias XHTML.

Para especificar donde queremos que se apliquen las plantillas utilizamos xsl:apply-templates.

El atributo match selecciona los nodos del árbol de entrada conforme a XPath, a los que se le va a aplicar la plantilla.

Con el atributo select seleccionamos los hijos del nodo de entrada conforme a XPath a los que se les aplica la plantilla.

Con ese atributo se puede especificar el orden en el que son procesados los nodos hijo, si no se especifica, se aplicará en el orden de arriba a abajo según se lee el documento XML.

Ejemplo

```
<xsl:template match="/">
<html>
  <body>
    <h2>Agenda de </h2>
    <xsl:apply-templates/>
  </body>
</html>
</xsl:template>
<xsl:template match="propietario/identificadores">
<h3>
<xsl:apply-templates select="apellidos"/>
,
<xsl:apply-templates select="nombre"/>
</h3>
</xsl:template>
```

El resultado:

Agenda de

López Terán , Alma

El Percebe 13, 6F Torrelavega 39300 970898765 942124567 628983456 Inés López Pérez El Ranchito 24, 6B Santander 39006 970123123 Roberto Gutiérrez Gómez El Marranito 4, 2F Santander 39004 970987456 942333323 Juan Sánchez Martínez El Cangrejo 10, sn Torrelavega 39300 997564343 942987974 677899234

5.1. Ejercicio resuelto de plantillas

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">

<xsl:template match="identificadores">
<xsl:value-of select="nombre"/>,
<xsl:value-of select="apellidos"/>
</xsl:template>

<xsl:template match="persona">
<xsl:apply-templates select="identificadores"></xsl:apply-templates>
</xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!DOCTYPE agenda>
<?xml-stylesheet type="text/xsl" href="./LMSGI_CONT_Ejemplo02.xsl"?>
<agenda>
<persona id="p01">
<identificadores>
<nombre>Inés</nombre>
<apellidos>López Pérez</apellidos>
</identificadores>
<direccion>
<calle>El Ranchito 24, 6B</calle>
<localidad>Santander</localidad>
<cp>39006</cp>
</direccion>

<telefonos>
<movil>970123123</movil>
</telefonos>
</persona>

<persona id="p02">
<identificadores>
<nombre>Roberto</nombre>
<apellidos>Gutiérrez Gómez</apellidos>
</identificadores>

<direccion>
<calle>El Marranito 4, 2F</calle>
<localidad>Santander</localidad>
<cp>39004</cp>
</direccion>

<telefonos>
<movil>970987456</movil>
<casa>942333323</casa>
</telefonos>
</persona>
</agenda>XML
```

Resultado

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

Inés,
López Pérez

Roberto,
Gutiérrez Gómez

Juan,
Sánchez Martínez
```

6. Procesadores XSLT

Es un software que lee un documento XSLT y otro XML y crea un documento de salida aplicando las instrucciones de la hoja de estilos XSLT a la información XML.

Puede estar integrado en el explorador web, servidor web o puede ser un programa desde la línea de comandos.

Existen diferentes modos de realizar la transformación XSLT:

- Mediante el procesador MSXML (servicios principales de Microsoft XML).
- Utilizando un procesador XSLTPROC como por ejemplo sltproc desde la línea de comandos.
- Invocando a la biblioteca de transformación desde un programa.
- Realizando un enlace entre la hoja XSLT y el documento XML, añadiendo en el fichero XML la definición de la versión XML y la definición del documento.

```
<?xml-stylesheet type=”text/xsl” href=”path_hoja_xsl”?>
```

La mayoría de editores XML permiten escoger el intérprete encargado de procesar un documento XSLT

7. Depuradores XSLT

Son elementos de software que permiten seguir la generación del documento a partir de un XML aplicándole hojas de estilo XSLT.

Nos facilitan la localización y corrección de errores dejando ejecutar código paso a paso.

Algunos editores XML incluyen un depurador que permite visualizar la plantilla y los datos.

8. Para saber más

- Recomendación del W3C sobre XSLT version 1.0.

- Recomendación del W3C sobre XSLT version 1.1.
- Recomendación del W3C sobre XSLT version 2.0.
- Recomendación del W3C sobre XSLT versión 3.0.
- MCLibre - teoría y ejemplos XPath

Mapa conceptual