Transformación de documentos XML con XSLT

Necesidad de las transformaciones

XML se presenta como un estándar para "transmitir" datos a través de Internet. Ante la posibilidad de que distintos "centros" o "aplicaciones" utilicen esquemas o DTD diferentes, es necesario un sistema que permita "transformar" los datos de un documento XML

XSLT (eXtensible Stylesheet Language – Transformations), describe un lenguaje basado en XML para transformar documentos XML a cualquier otro formato. Normalmente, utilizaremos XSLT para transformar documentos entre esquemas XML que permitan su procesamiento por distintos sistemas. También utilizaremos XSLT para transformar documentos XML en HTML, WML, o cualquier otro formato que facilite su presentación en la pantalla de un ordenador o en impresora. La transformación de XML a HTML es el principal uso que se hace de XSLT.

No debemos confundir las transformaciones XSLT con la presentación de documentos XML con CSS. Con XSLT, generaremos un documento HTML a partir de un documento XML. Se tratará de dos documentos "distintos". Con CSS, el navegador recibe un documento XML que formatea utilizando las reglas CSS para presentarlo en pantalla de forma que sea más fácilmente legible, pero es el mismo documento

XSLT, XSL, XSL FO…

XSLT es parte de la especificación XSL (eXtensible Stylesheet Language). En XSL se distingue entre:

- ✓ XSL FO (eXtensible Stylesheet Language Formatting Objects)
- ✓ XSLT (eXtensible StyleSheet Language Transformations), estable desde noviembre de 1999

XSL FO cuenta con escaso soporte por parte de la industria debido a su complejidad. Su propósito es definir la forma en la que se debe presentar un documento XML en papel o en pantalla. En este sentido, XSL FO sería una especificación similar a CSS.

Actualmente contamos con varias herramietas para realizar transformaciones XSLT:

- Saxon, desarrollado en Java por Michael Kay (un gurú de XSLT)
- xt, diseñado por James Clark
- Xalan un proyecto XML de Apache

Estructura de una hoja de estilo XSLT

Una hoja de estilo XSLT es un documento XML. Debe estar bien formado. Las hojas de estilo se guardarán siempre en archivos independientes con extensión .xsl que deben comenzar con una declaración XML:

```
<?xml version="1.0"?>
```

El elemento raíz de la hoja de estilo XSLT es stylesheet. Este elemento contendrá a todos los demás, y debe ir precedido por el alias xsl correspondiente al espacio de nombres para hojas de estilo XSLT. En las hojas de estilo XSLT, los nombres de los elementos "reservados" por la especificación, proceden de un mismo espacio de nombres, y por lo tanto deben escribirse precedidos por el correspondiente alias que debe "apuntar" a la URL: http://www.w3.org/1999/XSL/Transform De esta forma, el elemento raíz quedará así:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
.....................
</xsl:stylesheet>
```

Entre las marcas de inicio y de fin del elemento raíz xsl:stylesheet, se escribirán las reglas de transformación propiamente dichas definidas mediante un elemento xsl:template. La regla indica qué instancias de los elementos del documento XML se van a transformar, así como también indicará cómo se deben transformar cada una de ellas.

EJEMPLO:

```
<xsl:template match="//nombre">
   <h2>
      <xsl:value-of select="." />
   </h2>
</xsl:template>
```

La regla se aplicará a todas las instancias del elemento nombre. Esto se indica mediante el atributo **match** que acompaña al elemento **xsl:template**. Entre las etiquetas de inicio y de fin del elemento **xsl:template** se escribe la transformación que se debe realizar, es decir, **qué texto y qué marcas se escribirán en el documento resultado de la transformación**, cada vez que se encuentre una instancia del elemento **nombre** en el documento origen.

Con **<xsl:value-of...>**, se recupera y escribe en el documento resultado el valor del elemento que está siendo procesado.

Ejemplo de transformación XSLT

documento.xml

```
<?xml version="1.0"?>
<ciudades>
   <ciudad>
      <nombre>Madrid</nombre>
      <habitantes>3500000</habitantes>
   </ciudad>
   <ciudad>
      <nombre>Málaga</nombre>
      <habitantes>800000</habitantes>
   </ciudad>
   <ciudad>
      <nombre>Toledo</nombre>
      <habitantes>50000</habitantes>
   </ciudad>
</ciudades>
```

documento.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <head>
         <title>Ejemplo XSLT</title>
         </head>
         <body>
            <xsl:apply-templates select="nombre" />
         </body>
      </html>
   </xsl:template>
   <xsl:template match="//nombre">
      <h2>
         <xsl:value-of select="." />
      </h2>
   </xsl:template>
</xsl:stylesheet>
```

- ✓ La regla `<xsl:template match="/">` se ejecuta cuando se encuentra el elemento raíz del documento XML
- ✓ Dentro de esta regla, podemos incluir llamadas a otras reglas definidas en la hoja de estilo, mediante el elemento:

```
<xsl:apply-templates select="…" />
```

- ✓ El atributo `select` tomará como valor el nombre del elemento asociado a la regla que queremos "disparar"
- ✓ Esto nos ofrece un control real sobre el "orden" de ejecución de las reglas

Resultado de la transformación:

```
<html>
   <head>
      <title>Ejemplo XSLT</title>
   </head>
   <body>
      <h2>Madrid</h2>
      <h2>Málaga</h2>
      <h2>Toledo</h2>
   </body>
</html>
```

Ejercicio propuesto:

1. Con XML Spy, crear una primera hoja de estilo XSLT que, a partir del documento XML prestamos.xml, extraiga en un documento HTML el título de los libros que se han prestado
2. Formatear la lista de títulos para que se presente como una lista no ordenada (sin numerar), de HTML
3. Cambiar la hoja de estilo XSLT para que los títulos se presenten en una tabla con una única columna
4. Cambiar la hoja de estilo para que los títulos se presenten en una tabla con dos columnas. En la primera de ellas se escribirá un texto fijo: "LIBRO EN PRESTAMO"

El elemento <xsl:value-of…>

En el elemento `<xsl:value-of…>` se puede indicar que se quiere mostrar el valor del elemento que estamos procesando. También podemos indicar que queremos mostrar el valor de un elemento hijo, o descendiente, del elemento que se está procesando

En el ejemplo anterior, podríamos utilizar `xsl:value-of` para mostrar en el documento resultado de la transformación el título, código de registro o fecha de préstamo de cada libro…

Esto es posible porque en el atributo `select` podemos utilizar una "*expresión XPATH*". Por ejemplo, para mostrar el valor del elemento titulo, que es un hijo del elemento ejemplar, podríamos utilizar la siguiente regla:

```
   <xsl:template match="//ejemplar">
      <xsl:value-of select="./titulo" />
   </xsl:template>
```

El valor del atributo `select` se puede leer de la siguiente forma: "dame el valor del elemento ***titulo*** que es hijo del elemento que estoy procesando". En este caso, cada uno de los elementos ***ejemplar*** .Esto se indica mediante `./`

Resumen

En las reglas XSLT, entre sus marcas de inicio y de fin, se puede incluir:

- ✓ Texto que se escribirá "tal cual" en el documento resultado de la transformación.
- ✓ Marcas HTML o XML que se añadirán al documento resultado de la transformación.

- ✓ Elementos reservados de la especificación XSLT que realizarán una acción como recuperar el valor de un elemento, ordenar los resultados, llamar a otras reglas de la hoja de estilo, etc.

Orden de procesamiento

Las reglas se van activando y ejecutando a medida que se recorre el documento origen que se quiere transformar, de esta forma, las reglas se ejecutan en el orden en el que se van encontrando los elementos en el documento.

Este comportamiento por defecto puede cambiarse en las hojas de estilo XSLT, a diferencia de lo que sucedía en las hojas de estilo CSS. Esto permite "reordenar" los contenidos del documento XML, de una forma distinta a como están ordenadas en el documento XML inicial.

Para ordenar los contenidos, se utiliza el elemento `xsl:sort` que es un elemento hijo de `xsl:apply-templates`

Acepta dos atributos:

- ✓ `select` – que toma como valor el nombre del elemento que se va a utilizar como criterio de ordenación
- ✓ `order` – que indica si se debe utilizar un orden ascendente o descendente.

```
<xsl:apply-templates select="//ciudad">
   <xsl:sort select="ciudad" order="descending" />
</xsl:apply-templates>
```

Asociar una hoja de estilo a un documento

Debemos incluir, tras la declaración XML, la siguiente instrucción de procesamiento:

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

Ejemplo

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="http:/www.anaya.es/docs/xml/ejemplo.xsl"?>
<documento>
   <titulo>Programar ASP</titulo>
   <paginas>456</paginas>
   <anno-pub>2001</anno-pub>
</documento>
```

Leer y obtener el valor de atributos en XSLT

En XSLT podemos "filtrar" o indicar qué instancias de un elemento queremos procesar, tomando como criterio de selección el valor de los atributos que acompañan a los elementos.

Para hacer esto, en un elemento `xsl:value-of`, podemos recuperar el valor de un atributo mediante la expresión `@nombreAtributo` , por ejemplo:

```
<xsl:template match="vuelo">
   <tr>
      <td><xsl:value-of select="@numero" /></td>
      <td><xsl:value-of select="@origen" /></td>
      <td><xsl:value-of select="@destino" /></td>
      <td><xsl:value-of select="@hora" /></td>
   </tr>
</xsl:template>
```

Ejecución condicional de reglas

Para indicar qué instancias de un elemento queremos procesar, o realizar una "ejecución condicional de código", en XSLT disponemos del elemento `xsl:if` que va acompañado de un atributo test que contiene una "condición". Si la condición se cumple para el elemento que se está procesando, la regla se ejecutará. Por ejemplo:

```
<xsl:if test="@destino='JFK'">
<tr>
<td><xsl:value-of select="@numero" /></td>
<td><xsl:value-of select="@origen" /></td>
<td><xsl:value-of select="@destino" /></td>
<td><xsl:value-of select="@hora" /></td>
</tr>
</xsl:if>
```

xsl:choose, xsl:when y xsl:otherwise

Estos elementos "amplían" las posibilidades del elemento `xsl:if`. Permiten indicar qué transformación se debe realizar en el caso de que se cumpla una condición, y en el resto de casos.

Se utilizan de forma conjunta. El elemento `xsl:choose` contendrá a uno o más elementos `xsl:when` y a un elemento `xsl:otherwise`.

El elemento `xsl:when` incluye un atributo test que tomará como valor la expresión que se evaluará. Si se cumple, se ejecutará el código escrito entre las etiquetas de inicio y de fin del elemento `xsl:when`.

El elemento `xsl:otherwise` contendrá el código que se ejecutará si no se cumplen las expresiones indicadas en los atributos test de los elementos `xsl:when`.

```
<xsl:choose>
   <xsl:when test="expresión">
      .....
      .....
      .....
   </xsl:when>
   <xsl:when test="expresión2">
      .....
      .....
      .....
   </xsl:when>
   <xsl:otherwise>
      .....
      .....
   </xsl:otherwise>
</xsl:choose>
```

xsl:import y xsl:include

Es posible crear hojas de estilo XSLT modulares, es decir, divididas en distintos archivos físicos . En la hoja de estilo se incluirán referencias a otras hojas de estilo XSLT en las que se incluyen el resto de reglas.

Para incluir las referencias, se pueden utilizar los elementos `xsl:import` y `xsl:include`. Estos dos elementos deben ir acompañados por un elemento `href` que tomará como valor el URL absoluto o relativo de la hoja de estilo que se quiere utilizar.

Los elementos `xsl:import` se debe incluir justo a continuación de la etiqueta de inicio del elemento `xsl:stylesheet`, y antes de cualquier otro elemento.

El elemento `xsl:include` se puede incluir en cualquier lugar del documento, siempre que se escriba fuera de una regla `xsl:template`.

En cualquier hoja se podría incluir una referencia a otra hoja de estilo, utilizando la siguiente sintáxis:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:import href="hojaEstiloLibro.xsl" />
```

xsl:variable

El elemento `xsl:variable` se utiliza para declarar una variable. Las variables nos permiten realizar operaciones con los datos del documento XML para luego mostrar el resultado en el documento "resultado"

Es importante señalar que cuando se le asigna un valor, éste ya no se puede cambiar . Para declarar una variable, se utilizará la sintáxis:

```
<xsl:variable name="var" select="15" />
```

A continuación tenemos un ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="totalPrecio" select="sum(//total)" />
   <xsl:template match="/">
      <html>
      <head><title>Pedido</title></head>
      <body>
         <xsl:apply-templates />
      </body></html>
   </xsl:template>
   <xsl:template match="detalle">
      <table width="85%">
         <tr>
            <th>Material</th>
            <th>Unidades</th>
            <th>Precio</th>
            <th>Total Pts.</th>
         </tr>
         <xsl:for-each select="item">
         <tr>
            <td><xsl:value-of select="material" /></td>
            <td><xsl:value-of select="unidades" /></td>
            <td><xsl:value-of select="precio" /></td>
            <td><xsl:value-of select="total" /></td>
         </tr>
         </xsl:for-each>
      </table>
   <h4>Total a pagar: <xsl:copy-of select="$totalPrecio" /></h4>
   </xsl:template>
</xsl:stylesheet>
```

xsl:copy-of

Se utiliza para copiar un conjunto de nodos del documento origen, al documento resultado de la transformación. Se copiarán todos los nodos hijos y los atributos (en el caso de los elementos que los tengan).

Este elemento es especialmente útil cuando se quiere convertir un documento XML a otro documento XML con una estructura diferente. El elemento `xsl:copy-of` irá acompañado por un atributo `select` que toma como valor una expresión que determinará los nodos que se van a copiar.

Este elemento también se puede utilizar para copiar en el documento resultado el valor de una variable. En este caso, se escribirá como valor del atributo `select` el nombre de la variable precedido por el carácter `$`.

Ejemplo

Fichero xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="dlibros3.xsl"?>
<repertorio>
   <libro>
      <titulo>Don Quijote de la Mancha</titulo>
      <autor>Miguel de Cervantes</autor>
      <anno-pub>1987</anno-pub>
      <isbn>84-568-94-3</isbn>
   </libro>
```

```
   <libro>
      <titulo>La Galatea</titulo>
      <autor>Miguel de Cervantes</autor>
      <anno-pub>1989</anno-pub>
      <isbn>84-568-9424</isbn>
   </libro>
   <libro>
      <titulo>La Celestina</titulo>
      <autor>Fernando de Rojas</autor>
      <anno-pub>1998</anno-pub>
      <isbn>84-568-95-12</isbn>
   </libro>
</repertorio>
```

fichero xsl:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <repertorio>
      <xsl:copy-of select="//libro[starts-with(autor, 'Miguel de Cervantes')]" />
      </repertorio>
   </xsl:template>
</xsl:stylesheet>
```

fichero resultado:

```
<?xml version="1.0" encoding="UTF-8"?>
<repertorio>
   <libro>
      <titulo>Don Quijote de la Mancha</titulo>
      <autor>Miguel de Cervantes</autor>
      <anno-pub>1987</anno-pub>
      <isbn>84-568-94-3</isbn>
   </libro>
   <libro>
      <titulo>La Galatea</titulo>
      <autor>Miguel de Cervantes</autor>
      <anno-pub>1989</anno-pub>
      <isbn>84-568-9424</isbn>
   </libro>
</repertorio>
```

xsl:copy

Similar al elemento anterior, se utiliza para copiar elementos, pero no se copiarán sus atributos ni sus elementos hijos.

Cuando se aplica sobre elementos, se copia el elemento, pero no su valor.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <repertorio>
         <xsl:apply-templates select="//autor" />
      </repertorio>
   </xsl:template>
   <xsl:template match="autor">
      <xsl:copy />
   </xsl:template>
</xsl:stylesheet>
```

En este ejemplo, se crea un elemento autor vacío en el documento destino, para cada elemento autor existente en el documento original

Para copiar el valor de los elementos autor, habría que modificar la XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <repertorio>
         <xsl:apply-templates select="//autor" />
      </repertorio>
   </xsl:template>
   <xsl:template match="autor">
      <xsl:copy>
         <xsl:value-of select=”.” />
      </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

xsl:element

Se utiliza para crear elementos en el documento resultado de la transformación. Es especialmente útil cuando se utiliza XSLT para transformar un documento XML en otro con una estructura diferente.

`xsl:element` irá acompañado por un atributo `name` que tomará como valor el nombre del elemento que se va a crear. Si el elemento procede de un espacio de nombres, el URI que corresponde a este espacio de nombres se puede indicar en otro atributo: `namespace`

```
<xsl:template match=”div1”>
   <xsl:element name=”HTML:h1” namespace=”http://www.w3.org/HTML-transitional” />
</xsl:template>
```

xsl:attribute

Permite crear un atributo en el documento resultado de la transformación.

Irá acompañado por un atributo `name`, que recogerá el nombre del atributo, y opcionalmente por un atributo `namespace` que recogerá el alias del espacio de nombres del cual procede el atributo.

xsl:comment

Este elemento se utilizará para crear un comentario en el documento resultado de la transformación.

El elemento `xsl:comment` contendrá el texto del comentario, sin las marcas `<!--` y `-->`

xsl:processing-instruction

Se utiliza para crear una instrucción de procesamiento en el documento resultado de la transformación. Debe ir acompañado por un atributo `name`, que es obligatorio, y que toma como valor el nombre de la instrucción de procesamiento.

Entre sus etiquetas de inicio y de fin se escribirán los calificadores de la instrucción de procesamiento, entre las marcas `<xsl:text>` y `</xsl:text>`.

Ejemplo

El siguiente código crearía una instrucción de procesamiento en el documento destino:

```
<xsl:template match=”/”>
   <xsl:processing-instruction name=”xml-stylesheet”>
      <xsl:text>type=”text/xsl” href=”hojaEstilo.xsl”</xsl:text>
   </xsl:processing-instruction>
</xsl:template>
```