TRANSFORMACIÓN DE DOCUMENTOS (XSL)

CONTENIDOS

- Introducción
- Pasos para la transformación
- Las plantillas. Elemento XSL:TEMPLATE
- Instrucciones de control
- Uso de diferentes plantillas

Introducción

Tal y como se ha explicado en capítulos anteriores, la tecnología XML permite separar de manera efectiva los datos a almacenar, la estructura o semántica en la que se organizan y la presentación de los mismos.

Aunque podemos visualizar un fichero XML con un simple editor de texto, no es la manera más amigable para presenta los datos que están almacenados en su interior. De ahí donde debemos utilizar alguna herramienta que nos permita convertir y transformar los datos en el formato que deseamos. Esta herramienta se llama **XSL (*Extensible Stylesheet Languaje*).**

XSL es a XML lo que las hojas de estilo en cascada (CSS) a HTML. XSL permite tomar pleno control sobre los datos, pudiendo establecerse criterios como qué datos ver, en qué orden visualizarlos, establecer filtros y definir formatos de salida para su representación.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<libreria>
  <libro>
      <titulo>Physics-based animation</titulo>
      <autor>Kenny Erleben</autor>
      <editor>Charles River Media</editor>
      <isbn>978-1584503804</isbn>
      <precio>50.06</precio>
  </libro>
  <libro>
     <titulo>Principios de seguridad informática para
              usuarios</titulo>
      <autor>Carlos Garre</autor>
      <editor>Dykinson</editor>
      <isbn>978-84-9849-998-8</isbn>
      <precio>13.00</precio>
  </libro>
```

```
  <libro>
      <titulo>Ejercicios complementarios de lógica
              digital</titulo>
      <autor>Alberto Sánchez</autor>
      <editor>Dykinson</editor>
      <isbn>978-84-9849-703-8</isbn>
      <precio>12.00</precio>
  </libro>
</libreria>
```

Puede verse claramente que la librería puede contener un conjunto de libros. Cada libro tiene una serie de atributos propios y se han almacenado tres libros. Si queremos visualizarlo de una manera más amigable, podríamos definir una XSL como la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="/">
<html>
  <body>
      <h1> Mi biblioteca</h1>
      <table>
            <tr bgcolor="#887788">
                  <th>Título</th>
                  <th>Autor</th>
            </tr>
            <xsl:for-each select ="libreria/libro">
             <tr>
                  <td><xsl:value-of select="titulo" /></td>
                  <td><xsl:value-of select="autor" /></td>
            </tr>
            </xsl:for-each>
      </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

El resultado es:

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre
Ejercicios complementarios de lógica digital	Alberto Sánchez

En principio se puede transformar un documento XML a otro XL o a cualquier otro que distinto que pueda ser reconocido por un navegador web (HTML, XHTML).

Pasos para la transformación

Si se quiere utilizar correctamente esta teología de representación de documentos XML, se deben seguir los siguientes pasos:

1. Tener bien formado el documento XML.

2. Crear una hoja de estilo XSL bien formada. Usando el ejemplo anterior de los libros, se puede ver que el fichero XSL posee una declaración inicial que determina su contenido como XML y su codificación.

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

   A continuación se indica la etiqueta que identifica al fichero XSL y el espacio de nombres en el que se basa. En este caso tenemos:

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

3. Vincular al documento XML la hoja de estilo XSL.

   Para vincular en un documento XML una hoja de estilo XSL cualquiera, se debe escribir lo siguiente:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <?xml-stylesheet type="text/xsl"ref="tablaBiblio.xsl" ?>
   <libreria>
   ....
   </libreria>
   ```

Cumplidos los pasos anteriores, el procesador de documentos XLS comienza a recorrer el árbol del documento XML. Se inicia en el nodo raíz y se irá recorriendo todo el árbol haciendo que cada parte del documento XML pueda tener un formato especifico (árbol de resultados). Pero, ¿cómo determinar qué formatos tendrán cada una de las partes del documento?

La respuesta a la pregunta es la definición de plantillas en la hoja de estilos.

Las plantillas. Elemento XSL:TEMPLATE

Una plantilla es un patrón que cuando se cumple puede generar resultados de formateo en el árbol final. Es decir, con un documento de origen que se recorre por completo buscando concordancias con las plantillas definidas, de manera que si se cumple alguna, se aplica el formato definido y se genera en el árbol del documento final.

La siguiente etiqueta nos permite definir un elemento plantilla dentro de XSL. Todo lo que queda entre las siguientes etiquetas :

```
<xsl:template match="/">
…..
</xsl:template>
```

será lo que permitirá generar la salida formateada. Cabe destacar que con esta etiqueta se puede indicar sobre qué parte del documento XML se quiere actuar, utilizando el atributo ***match*** para ello. En este caso se ha querido actuar sobre la raíz del documento XML.

Instrucciones de control

Existen una serie de etiquetas que permiten reproducir el funcionamiento de las instrucciones de control de flujo de los lenguajes imperativos.

XSL:FOR-EACH

Permite iterar sobre una lista de elementos y aplicar transformaciones sobre ellos.

Atributos obligatorios:

- **select**: expresión que define la lista de elementos sobre los que va a iterar.

Las instrucciones <for-each> se pueden anidar para recorrer estructuras anidadas.

```
<xsl:for-each select ="libreria/libro">
```

XSL:VALUE-OF

Visualiza el contenido del elemento indicado en el atributo **`select`** y de todos sus descendientes.

Atributos obligatorios:

- **select**: expresión que especifica de qué elemento o atributo hay que extraer el contenido.

```
<xsl:value-of select="titulo" />
```

XSL:SORT

Envía a la salida datos del documento XML departida ordenados por algún criterio.

Atributos principales :

- **select**: expresión que especifica el nodo o conjunto de nodos por los que constituirán el criterio de ordenación.
- **order**: concreta si el orden es ascendente o descendente. Posibles valores: *ascending*, *descending*.
- **case-order:** especifica si las letras en minúsculas (*lower-first)* o mayúsculas (*upper-first*) aparecerán primero al ordenar.

```
<xsl:sort select=”autor”  order=”descending” />
```

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre
Ejercicios complementarios de lógica digital	Alberto Sánchez

Lo realmente interesante es que se puedan realizar búsquedas con determinados patrones preestablecidos. En este caso, parece lógico pensar que en vez de recorrerse todos los elementos, deberíamos filtrar por ese patrón (independientemente del orden que establezcamos). Si, por ejemplo, queremos extraer todos los libros del autor *"Kenny Erleben"*, se podría cambiar lo siguiente:

```
<xsl:for-each select ="libreria/libro[autor ='Kenny Erleben']" >
……
</xsl:for-each>
```

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben

Si por el contrario, lo que interesa es buscar con otro operador, como por ejemplo, todos los libros en los que el autor no es *"Carlos Garre"*, el cambio sería:

```
<xsl:for-each select ="libreria/libro[autor !='Carlos Garre']" >
…...
</xsl:for-each>
```

Mi biblioteca

Título	Autor
Ejercicios complementarios de lógica digital	Alberto Sánchez
Physics-based animation	Kenny Erleben

Hasta este momento, se han visto operadores de igualdad y desigualdad para cambiar el patrón de búsqueda pero no son los únicos que existen. Podrían utilizarse también los siguientes:

1. Operador de igualdad: =
2. Operador de desigualdad: !=
3. Operador menor que: <
4. Operador mayor que: >

XSL: IF

Se utiliza para producir un comportamiento condicional. Solo permite preguntar por una condición y actuar de una manera si la condición es cierta, es decir, no existe una rama del sino (else).

Atributo obligatorio:

- **test**: expresión que ha de hacerse cierta para que se aplique la plantilla que viene a continuación.

```
<xsl:for-each select ="libreria/libro">
 <xsl:sort select="autor" />
<xsl:if test="precio &gt; 12">
    <tr>
      <td><xsl:value-of select="titulo" /></td>
      <td><xsl:value-of select="autor" /></td>
    </tr>
</xsl:if>
</xsl:for-each>
```

XSL: CHOOSE, XSL: WHEN, XSL: OTHERWISE

El elemento *<xsl: if >* no es la única manera de condicionar los resultados. Existe otro elemento que nos permite establecer múltiples condiciones dentro del recorrido en el árbol XML. El elemento ***<xsl:choose>*** es muy similar a la instrucción *switch* de C/C++ o *case* de Pascal. Se pueden establecer tantas expresiones condicionales como se quieran mediante los elementos ***<xsl:when>***. Si se quiere establecer una condición por defecto, el elemento a utilizar sería ***<xsl:otherwise>***.

Siguiendo con el ejemplo inicial, establezcamos una condición nueva en la que si el libro es menor de 12.50€ muestre en color rojo todos sus datos. Si es mayor de 25.50€, en color verde. Y si no se cumple ninguna de las anteriores, en color azul.

```
<xsl:for-each select ="libreria/libro">
<tr>
      <xsl:choose>
            <xsl:when test="precio &lt; 12.50">
                  <td bgcolor="#ff0000">
                  <xsl:value-of select="titulo" />
                  </td>
                  <td bgcolor="#ff0000">
                  <xsl:value-of select="autor" />
                  </td>
            </xsl:when>
            <xsl:when test="precio &gt; 25.50">
                  <td bgcolor="#00ff00">
                  <xsl:value-of select="titulo" />
                  </td>
                  <td bgcolor="#00ff00">
                  <xsl:value-of select="autor" />
                  </td>
            </xsl:when>
            <xsl:otherwise>
                  <td bgcolor="#0000ff">
                  <xsl:value-of select="titulo" />
                  </td>
                  <td bgcolor="#0000ff">
```

```
                    <xsl:value-of select="autor" />
                    </td>
                </xsl:otherwise>
    </xsl:choose>
</tr>
</xsl:for-each>
```

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre
Ejercicios complementarios de lógica digital	Alberto Sánchez

Uso de diferentes plantillas

Hasta el momento solo hemos utilizado una única plantilla para todo el documento XML. Se habló del elemento ***<xsl:template>*** como etiqueta que permitía definir sobre qué parte del documento XML se quería actuar. El atributo ***match*** referenciaba siempre al elemento raíz del documento XML, no estableciendo distinción entre otras ramas o elementos que lo componen. A continuación mostramos un ejemplo usando diferentes plantillas para que podamos ver el nombre del autor en un color y el título del libro en otro :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="/">
<html>
   <body>
   <h1>Ejemplo plantillas</h1>
   <xsl:apply-templates />
   </body>
</html>
</xsl:template>

<xsl:template match="libreria">
   <h2>Mi biblioteca</h2>
     <table>
      <tr bgcolor="#887788">
          <th>Título</th>
          <th>Autor</th>
      </tr>
      <xsl:apply-templates select="libro"/>
     </table>
</xsl:template>

<xsl:template match="libro">
      <tr>
          <td><xsl:apply-templates select="titulo" /></td>
          <td><xsl:apply-templates select="autor" /></td>
```

```
        </tr>
    </xsl:template>

    <xsl:template match="titulo">
        <td bgcolor="#00FF00"><xsl:value-of select="." /></td>
    </xsl:template>

    <xsl:template match="autor">
        <td bgcolor="#A9F5A9"><xsl:value-of select="." /></td>
    </xsl:template>
</xsl:stylesheet>
```

Ejemplo plantillas

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre
Ejercicios complementarios de lógica digital	Alberto Sánchez

En este ejemplo se puede ver que se han establecido 5 plantillas personalizadas para los elementos **"/", "libreria", "libro", "autor"** y **"titulo".** Cuando se llame a esta hoja de estilos XSL desde un documento XML, se intentará reconocer los patrones de las plantillas sustituyendo en la salida los contenidos que se indiquen por cada plantilla.

Se debe destacar una serie de elementos que no se han explicado anteriormente y que permiten la aplicación directa de la plantilla en el documento final:

- **`<xsl:apply-templates />`** : Indica que apliquen el resto de plantillas definidas en cuanto se cumplan el patrón "***match***" de cada una.
- **`<xsl:apply-templates select="xxxx"/>`:** Indica que se aplique en ese momento el contenido de la plantilla **xxxx** definida en el documento XSL.
- **`<xsl:value-of select="." />`** : Indica que se copie el valor que contenga ese nodo. Por ejemplo, si el nodo actual es de tipo "titulo", insertará como resultado el título del libro actual.