

UT6. Transformación de documentos XML

UT6. Transformación de documentos

XML

6.1 Introducción

6.2. Técnicas de transformación de documentos XML

6.2.1. Diferencias entre las versiones de XSLT 1.0, 2.0 Y 3.0

6.3. Formatos de salida y ámbitos de aplicación

6.4. Filtrando contenido

6.5. Ordenando el contenido

6.6. Selección avanzada IF

6.7. Selección avanzada CHOOSE

6.8. Los atributos

6.1 Introducción

XSL es la sigla de extensible stylesheet language (lenguaje de hojas de estilo extensibles). Básicamente, es un lenguaje para la transformación de documentación. Se basa en los siguientes pilares:

- **Transformación.** Convirtiendo los ficheros XML en otro formato.
- **Formateo de objetos.** A través del diseño de un documento XML.
- **Búsqueda de contenido en documentos.** A través del árbol de nodos de un documento XML.

6.1 Introducción

El XSL es parte del XML y basa su versatilidad en tres componentes fundamentales:

1. **XPath.** Es un lenguaje para seleccionar o navegar entre los nodos en lenguaje XML dentro de un DOM.
2. **XSL-FO.** FO de formato de objetos (formatting objects). Transformaciones para convertir el documento XML en otro tipo de información como HTML o PDF.
3. **XSLT.** Sigla de XML stylesheets language for transformation (lenguaje de transformación basado en hojas de estilo).

6.1 Introducción



6.2. Técnicas de transformación de documentos XML

Una de las ventajas de las transformaciones es que pueden añadirse o quitarse elementos y atributos del fichero de salida. Como puede deducirse, XSLT ofrece mucho juego, puesto que los elementos de un fichero pueden reestructurarse, ordenarse, etc.

Además, pueden tomarse decisiones como qué elementos van a mostrarse u ocultarse, entre otras.



Básicamente, una transformación XSLT consiste en modificar un árbol de entrada en uno de salida.

6.2.1. Diferencias entre las versiones de XSLT 1.0, 2.0 Y 3.0

XSLT 1.0 fue creada el 16 de noviembre de 1999 y, ocho años después (en el 2007), nació XSLT 2.0. Por lo tanto, aunque son retro compatibles, la versión 2.0 tiene bastantes mejoras con respecto a la versión 1.0. La versión 3.0 salió el 2017 ampliando mucho más las capacidades de la versión 2.0.

Entre las novedades están las siguientes:

- Mejora del sistema de tipos. La versión 1.0 era muy restrictiva, puesto que solamente existían cinco: string, number, boolean, node-set y result tree fragment.
- Mejora de las funciones incorporadas y los operadores.
- Mejora del modelo de datos.
- Soporte para expresiones regulares.
- Múltiples árboles como resultado. En la versión 1.0, solamente podía crearse un árbol de salida.
- La instrucción `<xs:function>`. Sirve para poder definir funciones que puedan ser utilizadas en expresiones XPath.
- Mejora del sistema de errores.
- Mejora de la funcionalidad de agrupación. Por ejemplo, se añade la función `<xsl:for-each-group>`

6.2.1. Diferencias entre las versiones de XSLT 1.0, 2.0 Y 3.0

Una de las mejoras que se han incorporado es la posibilidad de streaming de los documentos fuente. Muchas veces, los documentos son muy grandes y se necesita producir resultados antes de que todo el documento esté disponible. Para ello, se crearon nuevas instrucciones para poder utilizar esta nueva funcionalidad como `<xsl:source-document>`, `<xsl:iterate>`, `<xsl:merge>` o `<xsl:fork>`.

Asimismo se añadió modularidad a los programas creando el concepto de paquete como en Java. Un paquete define un interfaz que regula funciones, variables, plantillas y otros componentes que, pese a estar visibles fuera del paquete, pueden ser sobrescritas. El objetivo último es la reutilización y mantenibilidad del código.

Otras instrucciones nuevas son:

- `xsl:evaluate`. Para evaluar expresiones XPath que se construyen dinámicamente con strings (o que se leen de un documento).
- `xsl:try`. Para recuperarse de errores en tiempo de ejecución.
- `xsl:global-context-item`. Se utiliza cuando se necesita un elemento cuyo contexto sea global (también se declara el tipo del elemento).
- `xsl:assert`. Para ayudar a los desarrolladores a crear código robusto.

6.3. Formatos de salida y ámbitos de aplicación

Como se ha mencionado anteriormente, XSLT puede transformar un documento XML en otro XML o en otro formato, como puede ser XHTML.

A continuación, se mostrará un ejercicio guiado para ver el formato de salida de una transformación XSLT, con los siguientes pasos.

1. Crear un fichero XML.
2. Crear un fichero XSL.
3. Abrir el fichero XML con un navegador.

6.3. Formatos de salida y ámbitos de aplicación

1. Crear un fichero XML.

```
raquetas.xml x raquetas.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="raquetas.xsl"?>
3 <TIENDA>
4 <RAQUETA>
5   <MARCA> BABOLAT </MARCA>
6   <MODELO> PURE DRIVE </MODELO>
7   <PRECIO> 200€ </PRECIO>
8 </RAQUETA>
9 <RAQUETA>
10  <MARCA> YONEX </MARCA>
11  <MODELO> MPTOUR 5 </MODELO>
12  <PRECIO> 150€ </PRECIO>
13 </RAQUETA>
14 <RAQUETA>
15  <MARCA> WILSON </MARCA>
16  <MODELO> HAMMER </MODELO>
17  <PRECIO> 1000€ </PRECIO>
18 </RAQUETA>
19 </TIENDA>
20
```

La estructura de este fichero es similar a la del documento XML visto anteriormente, pero la diferencia radica en la utilización de una plantilla XSL, como puede apreciarse mediante la siguiente línea:

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

Como puede observarse, la etiqueta indica que es una plantilla XSL y el atributo href con el valor "raquetas.xsl" indica la ubicación de la propia plantilla raquetas.xsl.

En este caso, se encuentra en la misma ubicación que el fichero XML.

6.3. Formatos de salida y ámbitos de aplicación

2. Crear un fichero XSL.

```
quetas.xml x raquetas.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3   <xsl:template match="/">
4
5     <html>
6     <body>
7       <h3>COLECCION DE RAQUETAS</h3>
8       <table border="1">
9         <tr>
10          <td> MARCA</td>
11          <td>MODELO</td>
12          <td>PRECIO</td>
13        </tr>
14        <xsl:for-each select="TIENDA/RAQUETA">
15          <tr>
16            <td><xsl:value-of select="MARCA"/></td>
17            <td><xsl:value-of select="MODELO"/></td>
18            <td><xsl:value-of select="PRECIO"/></td>
19          </tr>
20        </xsl:for-each>
21      </table>
22    </body>
23  </html>
24
25 </xsl:template>
26 </xsl:stylesheet>
27
```

En la orden siguiente del fichero XSL, pueden distinguirse dos partes:

<xsl:template match="/">

a) La propia orden <xsl:template>.

- Se utiliza para crear plantillas

b) El atributo match (match="/").

- Asocia una plantilla a un elemento XML.
- Normalmente, suele utilizarse para definir una plantilla para todo el documento XML. Para ello, se utiliza el atributo match con la expresión XPath "/", la cual indica que va a utilizarse todo el documento. En vez de la raíz, puede especificarse otra ruta en XPath.

El contenido dentro de la etiqueta <xsl:template> tendrá el HTML necesario para formatear la salida según la necesidad del programador.

Como se ha mencionado con anterioridad, el atributo match utiliza una expresión XPath, que se estudiará en profundidad más adelante.

6.3. Formatos de salida y ámbitos de aplicación

3. Abrir el fichero XML con un navegador.



COLECCION DE RAQUETAS

MARCA	MODELO	PRECIO
BABOLAT	PURE DRIVE	200€
YONEX	MPTOUR 5	150€
WILSON	HAMMER	1000€

6.3. Formatos de salida y ámbitos de aplicación

`<xsl:value-of>`

Dentro de la plantilla XSL del ejercicio anterior, pueden observarse órdenes del tipo:

```
<td><xsl:value-of select="MARCA" /></td>
<td><xsl:value-of select="MODELO" /></td>
<td><xsl:value-of select="PRECIO" /></td>
```

La etiqueta `<td>` ya se ha tratado, dado que es una etiqueta HTML, pero aparece una nueva etiqueta: `<xsl:value-of>`.

Esta etiqueta extrae el valor de un elemento XML y lo añade al flujo de salida de la transformación. En el ejemplo trabajado, a la salida HTML. Con el atributo `select`) el cual utiliza una expresión **XPath**, se seleccionará el elemento concreto.

6.3. Formatos de salida y ámbitos de aplicación

<xsl:for-each>

Otra etiqueta cuyo significado es intuitivo, pero que se estudiará a continuación, es la siguiente:

```
<xsl:for-each select="TIENDA/RAQUETA">
```

En esta etiqueta `<xsl:for-each>`, se seleccionarán todos los elementos uno a uno de una lista determinada. Los elementos que van a utilizarse estarán filtrados por el atributo `select`) el cual utilizará una expresión XPath.

6.4. Filtrando contenido

Cuando un fichero es muy grande, lo más normal es filtrar el contenido y no trabajar con todo el árbol de nodos. Por ejemplo, trabajar solamente con las raquetas de la marca BABOLAT.

La etiqueta `<xsl:for-each>` podría modificarse de la siguiente forma para que filtre solamente las raquetas de dicha marca:

```
<xsl:for-each select= "TIENDA/RAQUETA[MARCA='BABOLAT']">
  <tr>
    <td><xsl:value-of select="MARCA" /></td>
    <td><xsl:value-of select="MODELO" /></td>
    <td><xsl:value-of select="PRECIO" /></td>
  </tr>
</xsl:for-each>
```

6.5. Ordenando el contenido

En este momento, se necesita que, además de la marca BABOLAT, las raquetas queden ordenadas por el precio.

Es preciso utilizar la etiqueta `<xsl:sort>` añadiendo la propiedad `select="PRECIO"` . El resultado sería añadir la siguiente línea al fichero:

```
<xsl:sort select= "PRECIO"/>
```

Véase cómo quedaría el filtro anterior incluyendo la ordenación:

```
<xsl:for-each select= "TIENDA/RAQUETA[MARCA='BABOLAT']">
  <xsl:sort select= "PRECIO" />
  <tr>
    <td><xsl:value-of select="MARCA" /></td>
    <td><xsl:value-of select="MODELO" /></td>
    <td><xsl:value-of select="PRECIO" /></td>
  </tr>
</xsl:for-each>
```

Puede usarse también `order="descending"` para la ordenación inversa.

6.6. Selección avanzada IF

Si quieren mostrarse todas las raquetas con un precio superior a 100 euros. Para esta operación, puede utilizarse la etiqueta `<xsl:if>` de la siguiente forma:

```
<xsl:if test="PRECIO>100">
```

El lugar donde hay que colocar esta sentencia es siempre dentro de la etiqueta `<xsl:for-each>`.

Véase el resultado completo:

```
<xsl:for-each select= "TIENDA/RAQUETA[MARCA='BABOLAT']">
  <xsl:if test= "PRECIO>100" />
    <tr>
      <td><xsl:value-of select="MARCA" /></td>
      <td><xsl:value-of select="MODELO" /></td>
      <td><xsl:value-of select="PRECIO" /></td>
    </tr>
  </xsl:for-each>
```

Ojo que se debería quitar el € de los precios del ejemplo.

6.7. Selección avanzada CHOOSE

Imagínese que quiere conseguirse un resultado como el siguiente:

COLECCION DE RAQUETAS

MARCA	MODELO	PRECIO
BABOLAT	PURE DRIVE	200
YONEX	MPTOUR 5	150
WILSON	HAMMER	1000

Dependiendo del valor de la raqueta, el precio se sombreadá de amarillo si supera el preciode 150 euros y de verde en caso contrario.

6.7. Selección avanzada CHOOSE

La plantilla que habrá que utilizar para realizar esta transformación es la siguiente:

Se ha utilizado la etiqueta `<xsl:choose>` de tal manera que, al seleccionar las raquetas con precio mayor a 150 euros:

```
<xsl:when test="PRECIO>150">
```

La salida tendrá un sombreado y, en caso contrario:

```
<xsl:otherwise>
```

El sombreado será diferente.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3   <xsl:template match="/">
4
5     <html>
6       <body>
7         <h3>COLECCION DE RAQUETAS</h3>
8         <table border="1">
9           <tr>
10            <td> MARCA </td>
11            <td> MODELO </td>
12            <td> PRECIO </td>
13          </tr>
14          <xsl:for-each select="TIENDA/RAQUETA">
15            <tr>
16              <td><xsl:value-of select="MARCA"/></td>
17              <td><xsl:value-of select="MODELO"/></td>
18              <xsl:choose>
19                <xsl:when test="PRECIO>150">
20                  <td bgcolor="#fff66"><xsl:value-of select="PRECIO"/></td>
21                </xsl:when>
22                <xsl:otherwise>
23                  <td bgcolor="#99ff99"><xsl:value-of select="PRECIO"/></td>
24                </xsl:otherwise>
25              </xsl:choose>
26            </tr>
27          </xsl:for-each>
28        </table>
29      </body>
30    </html>
31
32  </xsl:template>
33 </xsl:stylesheet>
34
```

6.8. Los atributos

La instrucción `<xsl:attribute>` permite generar un atributo y su valor. Se utiliza cuando el valor del atributo se obtiene a su vez de algún nodo.

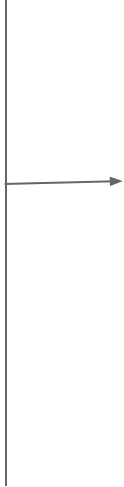
Por ejemplo, a partir de un documento XML, se quiere generar la etiqueta `` en la que el valor del atributo `src` sea el contenido de la etiqueta `<imagen>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<licencias>
  <licencia>
    <nombre>Creative Commons By - Share Alike</nombre>
    <imagen>cc-bysa-88x31.png</imagen>
  </licencia>
</licencias>

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

  <xsl:template match="licencia">
    <p><img>
      <xsl:attribute name="src">
        <xsl:value-of select="imagen" />
      </xsl:attribute>
    </img>
    </p>
  </xsl:template>

</xsl:stylesheet>
```



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