Hojas de Estilo XSLT

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.

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

XSLT es un **lenguaje declarativo**. Por ello, las hojas de estilo XSLT no se escriben como una secuencia de instrucciones, sino como una colección de plantillas (*template rules*). Cada plantilla establece **cómo se transforma un determinado elemento** (definido mediante expresiones XPath). La transformación del documento se realiza de la siguiente manera:

- Se analiza el documento y construye el árbol del documento.
- Se va recorriendo todos los nodos desde el nodo raíz, aplicando a cada nodo una plantilla, sustituyendo el nodo por el resultado.
- Cuando se ha recorrido todos los nodos, se ha terminado la transformación.

Una hoja de estilo XSLT es un documento XML que contiene al menos las etiquetas siguientes:

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

Estas etiquetas son:

- La **declaración** xml **`<?xml>`**, propia de cualquier documento XML.
- La **instrucción `<xsl:stylesheet>`** es la etiqueta raíz de la hoja de estilo, y sus atributos indican la versión y el espacio de nombres correspondiente.

Dentro de la instrucción **`<xsl:stylesheet>`** se pueden encontrar los llamados **elementos de alto nivel** y las **plantillas**, como en el ejemplo siguiente:

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

Estas etiquetas son:

- El elemento de alto nivel **<xsl:output>** indica el tipo de salida producida.
- La instrucción **<xsl:template>** es una plantilla.
    - El **atributo match** indica los elementos afectados por la plantilla y contiene una expresión XPath.
    - El contenido de la instrucción define la transformación a aplicar (si la instrucción no contiene nada, como en el ejemplo anterior, sustituirá el nodo por nada).

Pasos:

1. Tener bien definido el documento xml.
2. Crear una hoja de estilo xsl bien formada.
3. Vincular al documento xml la hoja de estilo xsl.

1. Tener bien definido el documento xml

Libreria.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<libreria>
  <libro>
      <titulo>Physics-bases animation</titulo>
      <autor>Kennu 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>
```

2. Crear una hoja de estilo xsl bien formada

<xsl:template match="/"> permite definir un elemento plantilla dentro del XSL. Todo lo que quede 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 propio documento xml.

Libreria.xsl

```
<?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 style="background-color:#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>
```

3. Vincular al documento xml la hoja de estilo xsl.

Ahora vinculamos.

Libreria.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="libreria.xsl"?>
<libreria>
  <libro>
      <titulo>Physics-bases 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>
```

Al abrir el fichero ***librería.xml*** con el navegador:

Elementos básicos

xsl:for-each

Es posible que al usuario le interese recorrer todos y cada uno de los libros y extraer los datos para darles un nuevo formato. En este caso, la siguiente etiqueta indica que se recorran todo el conjunto de elementos xml que sean libros.

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

xsl:value-of

Extrae el contenido del libro

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

xsl:sort

La muestra de forma ordenada. Si lo queremos ordenar por **titulo** tras la etiqueta `for-each` incluimos:

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

Si el orden es por autor:

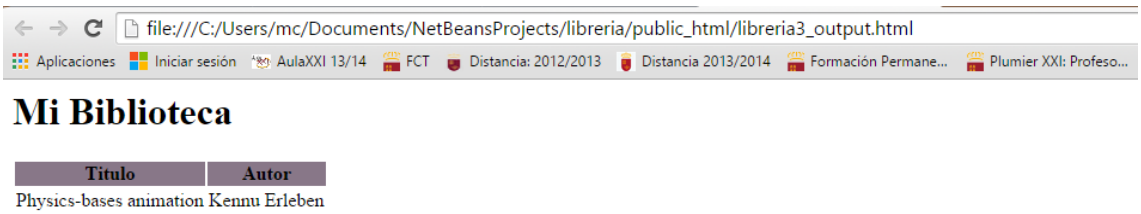
```
<xsl:sort select="autor"/>
```

Ejercicio: Ordenado por titulo

Ejercicio: Ordenado por autor

Ejercicio: Se podría filtrar por una expresión, independiente del orden que establezcamos. Por ejemplo, extraer todos los libros del autor “Kenny Erleben”

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



Operadores para construir las expresiones de búsqueda.

Operador	Significado
=	Igualdad
!=	Desigualdad
<	Menor que
>	Mayor que
<=	Menor o igual
>=	Mayor o igual
not	no
and	y
or	o

Ejercicio: Extraer todos los libros que no pertenezcan al autor 'Carlos Garre'

Mi Biblioteca

Titulo	Autor
Ejercicios complementarios de lógica digital	Alberto Sánchez
Physics-bases animation	Kenny Erleben

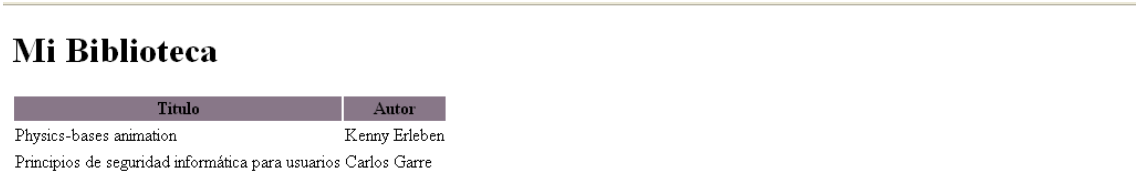
<xsl:if>

También existe la posibilidad de indicar con el elemento `<xsl:if>` condiciones más complejas en la evaluación del fichero XML.

La sintaxis de este nuevo elemento es la siguiente:

```
<xsl:if test="expresion">
```

Ejercicio: imaginemos que queremos mostrar todos aquellos libros cuyo coste sea superior a 12 euros.



Mi Biblioteca

Titulo	Autor
Physics-bases animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre

<xsl:choose>

`<xsl:choose>` nos permite establecer múltiples condiciones dentro del recorrido en el árbol XML. Es una instrucción similar a switch en C, Java... Se pueden establecer tantas expresiones condicionales como quieran mediante los elementos `<xsl:when>`. Si se quiere establecer una condición por defecto, el elemento a utilizar sería `<xsl:otherwise>`.

Sintaxis:

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

Ejercicio: Visualizar los libros: los datos en color rojo si el precio es menor de 12.50 euros, en verde si es mayor de 25.50 euros y en azul si no cumple ninguna de las anteriores.

Mi Biblioteca

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

Si ahora ***Librería.xml***

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="probando2.xsl"?>

<libreria>
  <libro id="1">
      <titulo  >Physics-bases animation</titulo>
      <autor>Kenny Erleben</autor>
      <editor>Charles River Media</editor>
      <isbn>978-1584503804</isbn>
      <precio moneda="euros">50.06</precio>
      <foto>logotipo.jpg</foto>
  </libro>
  <libro id=”2”>
      <titulo >Principios de seguridad informática para
usuarios</titulo>
      <autor>Carlos Garre</autor>
      <editor>Dykinson</editor>
      <isbn>978-84-9849-998-8</isbn>
      <precio moneda="dolares">13.00</precio>
      <foto>informatica3.jpg</foto>
  </libro>
  <libro  id="3">
      <titulo  >Ejercicios complementarios de lógica
digital</titulo>
      <autor>Alberto Sánchez</autor>
      <editor>Dykinson</editor>
      <isbn>978-84-9849-703-8</isbn>
      <precio moneda="euros">12.00</precio>
      <foto>informatica2.jpg</foto>
    </libro>
</libreria>
```

Seleccionar por atributos: @nombreDelAtributo

Ejemplo:

```
<xsl:for-each select="libreria/libro[@id='1']">
```

Visualizar un atributo

Ejemplo:

```
<xsl:value-of select="precio/@moneda”/>
```

Convertir el valor de un elemento en el valor de un atributo:

```
<xsl:attribute name="src">
     <xsl:value-of select="foto" />
</xsl:attribute>
```

Ejemplo

```
<td>
  <img>
   <xsl:attribute name="src">
    <xsl:value-of select="foto" />
   </xsl:attribute>
  </img>
</td>
```

Las plantillas <xsl:template>

Hasta ahora 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. Si se observan con detenimiento todos los ejemplos anteriores, el atributo `match` referenciaba siempre la raíz del documento xml, no estableciendo distinción entre otras o elementos que lo componen.

Cuando se aplica una plantilla a un nodo, en principio la plantilla se aplica únicamente al nodo, pero se sustituye el nodo y todos sus descendientes por el resultado de la aplicación de la plantilla, lo que nos haría perder a los descendientes. Si se quiere que antes de sustituir el nodo y todos sus descendientes se apliquen también a los descendientes las plantillas que les correspondan, hay que utilizar la instrucción **<xsl:apply-templates />,** como en el ejemplo 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='/'>
      <xsl:apply-templates />
    </xsl:template>

    <xsl:template match="elemento">
      ...
    </xsl:template>

  </xsl:stylesheet>
```

```
  <?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>
        <xsl:apply-templates />
      </html>
    </xsl:template>

    <xsl:template match="elemento">
     ...
    </xsl:template>

  </xsl:stylesheet>
```

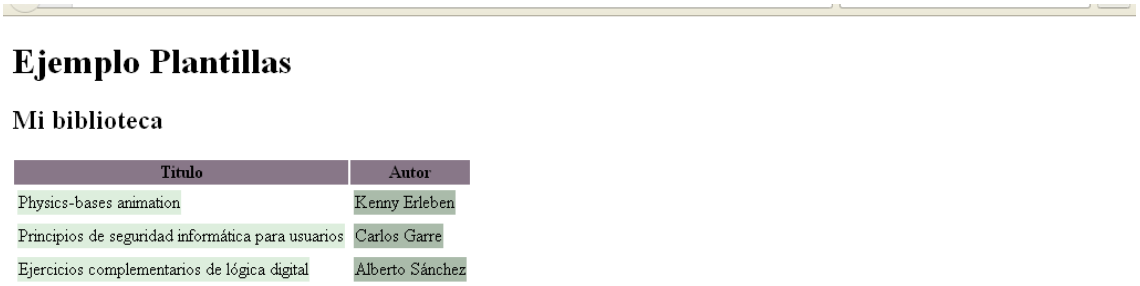
Se debe destacar una serie de elementos que permiten la aplicación directa de la plantilla en el documento final.

<xsl:apply-templates/> indica que se apliquen el resto de las 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="."/> En el nodo xml actual de la plantilla actual, 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.

Ejercicio: Se pretende establecer cinco plantillas personalizadas para los elementos "/" "librería" "libro" "titulo" "autor". 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.



Ejemplo Plantillas

Mi biblioteca

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

```
      <?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>Titulo</th>
              <th>Autor</th>
            </tr>
            <xsl:apply-templates select="libro"/>
          </table>
        </xsl:template>

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

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

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

      </xsl:stylesheet>
```