1 INTRODUCCIÓN A XSL

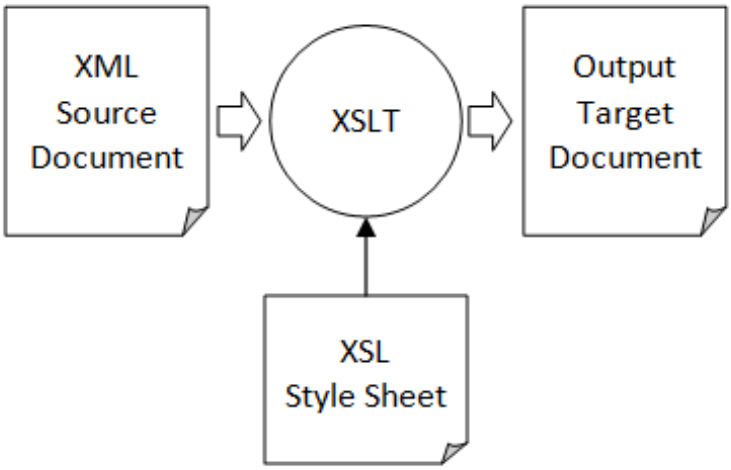
XSL (eXtensible Stylesheet Language, lenguaje extensible de hojas de estilo) es una familia de lenguajes XML para expresar hojas de estilo (cómo debe representarse un XML concreto en un medio). Consiste en tres componentes principales:

- XSLT, un lenguaje de transformación de documentos.
- XPath, un lenguaje para referenciar a partes de documentos XML.
- XSL-FO, un lenguaje de formato de documentos XML.

XSLT (eXtensible Stylesheet Language for Transformations) es un lenguaje que permite aplicar una transformación a un documento XML para obtener otro documento XML, un documento HTML o un documento de texto plano.

La hoja de estilos XSLT con las reglas de transformación es también un documento de texto XML en sí, generalmente con extensión .xsl, por lo que se podrá comprobar si está bien formado o no.

El funcionamiento lo podemos observar en las imágenes:



A un documento XML se le pueden aplicar una o varias transformaciones XSLT e incluso una transformación CSS. Las hojas de estilos XSLT son más útiles que las hojas de estilos CSS porque:

- Permiten cambiar el orden los elementos.
- Permiten realizar operaciones con sus valores.
- Permiten agrupar elementos.

De ahí que se suelan utilizar en combinación más que decantarse por una u otra hoja de estilos.

2 ENLAZAR XML CON XSL

Para indicar que un documento XML se debe transformar con una hoja de estilo XSL lo indicaremos al incicio del documento XML

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

Cuando abrimos el XML con un navegador realizará la transformación.

Nota: Chrome, por seguridad, no realiza la transformación cuando abrimos el archivo a través del sistema de ficheros (C:/... o /home/..), solo lo hace si accedemos a él a través de un servidor (http://localhost.. o http://www...)

3 EL ELEMENTO XSL:TEMPLATE

Una hoja de estilo de XSL consiste en una serie de plantillas (*template*s) de transformación. Cada elemento xsl:template contiene las transformaciones que XSL debe aplicar si el patrón especificado en el elemento coincide con lo encontrado en el documento XML.

Para especificar el elemento XML al que debemos aplicar el *template* utilizaremos el atributo match (también podemos aplicar la plantilla a todo el documento XML, para lo cual podemos especificar match="/").

Los valores que podemos asignar al atributo match son los especificados por el estándar XPath.

Por ejemplo, la siguiente transformación XSL devuelve un código XHTML concreto al procesar el documento con el expediente del alumno.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Expediente académico</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th align="left">Asignatura</th>
            <th align="left">Nota</th>
          </tr>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Como podemos ver, si probamos este documento XSL, el resultado es tan sólo una cabecera de página. Si analizamos el documento XSL, veremos que dispone de una plantilla que se aplica cuando coincide con el elemento raíz del documento (match="/") y que imprime al resultado lo que está contenido en la etiqueta.

4 EL ELEMENTO XSL:VALUE-OF

El elemento value-of se usa para seleccionar y añadir a la salida el valor del elemento XML seleccionado.

Por ejemplo, si añadimos el siguiente código a nuestro ejemplo anterior:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Expediente académico</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th align="left">Asignatura</th>
            <th align="left">Nota</th>
          </tr>
          <tr>
            <td><xsl:value-of select="expediente/asignatura/nombre"/></td>
            <td><xsl:value-of select="expediente/asignatura/nota"/></td>
          </tr>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Veremos en el resultado que aparece la primera nota del expediente.

Esto se debe a que las etiquetas value-of seleccionan el valor del primer elemento que cumple con el patrón especificado.

5 EL ELEMENTO XSL:FOR-EACH

El elemento xsl:for-each de XSL puede utilizarse para seleccionar cada uno de los elementos del documento XML que pertenezcan a un conjunto determinado.

Si al ejemplo anterior, donde sólo aparecía la primera nota del expediente, le añadimos un xsl:for-each que realice el recorrido por todo el expediente de la siguiente forma:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
    <body>
      <h2>Expediente Académico</h2>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th align="left">Asignatura</th>
          <th align="left">Nota</th>
        </tr>
        <xsl:for-each select="expediente/asignatura">
          <tr>
            <td><xsl:value-of select="nombre"/></td>
            <td><xsl:value-of select="nota"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Obtendremos ahora un listado de todas las notas de las asignaturas.

6 ORDENACIÓN DE LA INFORMACIÓN CON XSL:SORT

Para obtener una salida ordenada, simplemente debemos añadir un elemento xsl:sort al elemento xsl:for-each en nuestro fichero XSL:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Expediente Académico</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th align="left">Asignatura</th>
            <th align="left">Nota</th>
          </tr>
          <xsl:for-each select="expediente/asignatura">
            <xsl:sort select="nombre"/>
            <tr>
              <td><xsl:value-of select="nombre"/></td>
              <td><xsl:value-of select="nota"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

El atributo select nos permite indicar el elemento por el que vamos a realizar la ordenación, en este caso por nombre de asignatura.

Los siguientes atributos se pueden utilizar con el elemento xsl:sort:

- lang: "language-code"
- data-type: "text | number | qname"
- order: "ascending | descending"
- case-order: "upper-first | lower-first"

Ejemplo:

<xsl:sort select="nombre" lang="es" data-type="text" order="ascending" case-order="upper-first"/>

7 CONDICIONES EN XSL

Disponemos de dos elementos XSL que nos permiten implementar condiciones en nuestras transformaciones. Se trata de xsl:if y xsl:choose.

7.1 EL ELEMENTO XSL:IF

El elemento xsl:if nos permite aplicar una plantilla sólo en el caso de que la condición especificada se cumpla (sea cierta).

<xsl:if test="nota < 5">

…sólo aparecerá con nota menor a 5…

</xsl:if>

Por ejemplo, podemos modificar el código anterior para que sólo muestre las notas superiores a 5.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Expediente Académico</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th align="left">Asignatura</th>
            <th align="left">Nota</th>
          </tr>
          <xsl:for-each select="expediente/asignatura">
            <xsl:if test="nota > 5">
              <tr>
                <td><xsl:value-of select="nombre"/></td>
                <td><xsl:value-of select="nota"/></td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

7.2 EL ELEMENTO XSL:CHOOSE

El elemento xsl:choose (junto con xsl:when y xsl:otherwise), nos permite modelar tests condicionales múltiples. Esto es, podemos, en función de una condición múltiple (con múltiples valores posibles), obtener resultados diversos.

Un ejemplo de formato de xsl:choose es el siguiente:

<xsl:choose>

<xsl:when test="nota < 5">

…código (suspendido)…

</xsl:when>

<xsl:when test="nota < 9">

…código (normal)…

</xsl:when>

<xsl:otherwise>

…código (excelente)…

</xsl:otherwise>

</xsl:choose>

Modificamos el ejemplo anterior para que las notas inferiores a cinco aparezcan en color rojo.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Expediente Académico</h2>
        <table border="1">
          <tr bgcolor="#9acd32">
            <th align="left">Asignatura</th>
            <th align="left">Nota</th>
          </tr>
          <xsl:for-each select="expediente/asignatura">
            <tr>
              <td><xsl:value-of select="nombre"/></td>
              <td>
                <xsl:choose>
                  <xsl:when test="nota < 5">
                    <font color="#FF0000">
                      <xsl:value-of select="nota"/>
                    </font>
                  </xsl:when>
                  <xsl:otherwise>
                    <xsl:value-of select="nota"/>
                  </xsl:otherwise>
                </xsl:choose>
              </td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

8 EL ELEMENTO XSL:APPLY-TEMPLATES

apply-templates aplica una plantilla al elemento actual o a los elementos hijos del elemento actual. Con el atributo select podemos procesar sólo aquellos elementos hijos que especifiquemos y el orden en que deben procesarse.

Por ejemplo:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>Expediente académico</h2>
        <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>

  <xsl:template match="expediente">
    <xsl:apply-templates select="asignatura"/>
  </xsl:template>

  <xsl:template match="asignatura">
    <p>
      <xsl:apply-templates select="nombre"/>
      <xsl:apply-templates select="nota"/>
    </p>
  </xsl:template>

  <xsl:template match="nombre">
    Nombre: <xsl:value-of select="."/>
    <br/>
  </xsl:template>

  <xsl:template match="nota">
    Nota: <xsl:value-of select="."/>
    <br/>
  </xsl:template>
</xsl:stylesheet>
```

Como podemos ver, esta organización es mucho más modular y nos permite un mejor mantenimiento y revisión de la hoja de estilo.

9 CREACIÓN DE NUEVOS NODOS CON XSL:ELEMENT Y XSL:ATTRIBUTE

Cuando tengamos que crear nodos nuevos en el documento de salida (con sus etiquetas y atributos), necesitaremos utilizar los elementos xsl:element y xsl:attribute para poder utilizar los valores del XML origen en dichos nodos.

En el siguiente ejemplo crearemos un enlace HTML, es decir, una etiqueta "a" con el atributo "href" utilizando el atributo "url" de un XML.

XML inicial: *<enlace url="www.google.es">Google</enlace>*
XSL transformación: *<xsl:element name="a">* *<xsl:attribute name="href">* *<xsl:value-of select="@url" />* *</xsl:attribute>* *</xsl:element>*
HTML salida: *<a href="www.google.es" />*