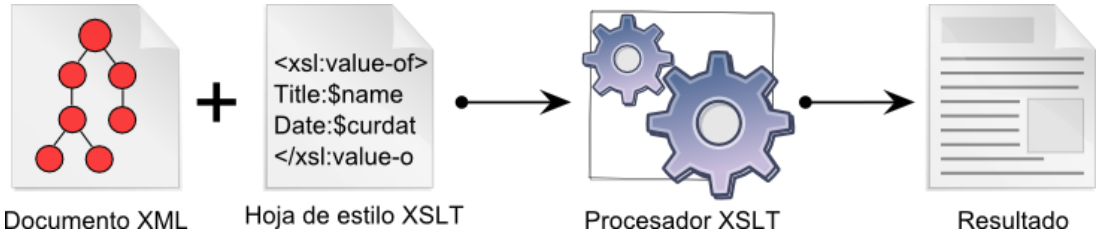
# XSLT

XSLT (Transformaciones XSL) es un lenguaje de programación declarativo que permite generar documentos a partir de documentos XML, como ilustra la imagen siguiente:




- El documento XML es el documento inicial a partir del cual se va a generar el resultado.
- La hoja de estilo XSLT es el documento que contiene el código fuente del programa, es decir, las reglas de transformación que se van a aplicar al documento inicial.
- El procesador XSLT es el programa de ordenador que aplica al documento inicial las reglas de transformación incluidas en la hoja de estilo XSLT y genera el documento final. Lo hace de la siguiente manera:

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

- El resultado de la ejecución del programa es un nuevo documento (que puede ser un documento XML o no).

XSLT se utiliza para obtener a partir de un documento XML otros documentos (XML o no). A un documento XML se le pueden aplicar distintas hojas de estilo XSLT para obtener distintos resultados y una misma hoja de estilo XSLT se puede aplicar a distintos documentos XML.

Ha habido dos versiones:

- noviembre de 1999: XSLT 1.0
- enero de 2007: XSLT 2.0

# COMO ENLAZAR DOCUMENTOS

Para realizar una transformación XSLT necesitamos un documento .xml y un documento .xsl. Ambos documentos son xml, luego tienen que estar bien formados.

Nosotros vamos a trabajar con documentos XSL con esta forma:

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

o esta

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

Al documento XML del cual extraemos los datos
tenemos que añadirle esta línea donde enlazamos con el XSLT correspondiente:

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

Ver ejemplo miscds.xml con miscds.xsl, simplemente abriendo miscds.xml.

Al abrir miscds.xml con el navegador mozilla o explorer se ve el archivo ya transformado. Esto a veces no funciona en local por problemas de seguridad.
Otra forma de ver la transformación es en:
https://freeformatter.com/xsl-transformer.html

# ELEMENTOS

Dentro de un template se pueden poner etiquetas html y otros elementos que pasamos a definir a continuación:

## Elemento <xsl:value-of>

Se usa para extraer el valor de un nodo determinado.
Tiene un atributo **select** con una expresión XPath.

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

Muestra el title de un elemento. Previamente se ha tenido que posicionar en él.

## Elemento <xsl:for-each>

Sirve para hacer bucles, para seleccionar elementos. Selecciona todos los elementos de un conjunto de nodos.
Su atributo **select** tiene una expresión XPath. Este ejemplo nos posicionaría en todos los elementos cd hijos de catalog:

```
<xsl:for-each select="catalog/cd">
```

Se puede filtrar el resultado, por ejemplo mostrando los cds cuyo artista es Bob Dylan:

```
<xsl:for-each select=”catalog/cd[artist=’Bob Dylan’]”>
```

## Elemento <xsl:sort>

Se usa para ordenar la salida.
Se coloca dentro de la etiqueta <xsl:for-each>
Tiene un atributo **select** que indica el elemento por el que se realiza la ordenación.
Nota: buscar resto de atributos en w3schools.
Ejemplo: ordena los cds por artista.

```
<xsl:for-each select="catalog/cd">
    <xsl:sort select="artist"/>
```

## Elemento <xsl:if>

Tiene un atributo **test=”condicion”** que se tiene que cumplir para que se realice el output.
Ejemplo: Muestra los precios iguales a 9.90.

```
    <xsl:if test="price = 9.90">
            <xsl:value-of select="price"/>
   </xsl:if>
```

Estos if no tienen else.

Otros posibles operadores son:

<  menor que
<= menor o igual que
> mayor que
>= mayor o igual que
!= distinto
and  y lógico
or  o lógico

## Elemento <xsl:choose>

Sirve para hacer una elección múltiple.
Esta es su sintaxis:

```
<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```

Ejemplo: Pone el fondo de un color a los artistas cuyos precios son mayores de 10, de otro a los que el precio es menor que 9 .

```
<xsl:for-each select="catalog/cd">
   <tr>
     <td><xsl:value-of select="title"/></td>
<xsl:choose>
       <xsl:when test="price &gt; 10">
         <td bgcolor="#ff00ff">
         <xsl:value-of select="artist"/></td>
       </xsl:when>
       <xsl:when test="price &lt; 9">
         <td bgcolor="#cccccc">
         <xsl:value-of select="artist"/></td>
       </xsl:when>
       <xsl:otherwise>
         <td><xsl:value-of select="artist"/></td>
       </xsl:otherwise>
     </xsl:choose>
```

## Elemento xsl:atribute

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 del siguiente documento xml, se quiere generar la etiqueta *<img>*. 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>
```

Este documento xslt:

```
    <?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>
   <p><img>
  <xsl:for-each select="//licencia">
   <xsl:attribute name="src">
    <xsl:value-of select="imagen" />
   </xsl:attribute>
   </img>
   </p>
</html>
</xsl:template>
</xsl:stylesheet>
```

Provoca la siguiente salida:

```
<html>
<p>
 <img src="cc_bysa_88x31.png"/>
</p>
</html>
```

## Elemento <xs:variable>

En xslt una variable realmente es una constante. Consiste en dar un valor a un identificador que se podrá usar más adelante usando $ por delante.
Nos creamos una variable que asigna el color rojo

```
<xsl:variable name="color" select='"red"' />
o así
<xsl:variable name=”color”>red</xsl:variable>
```

Luego la podemos usar asi:

```
<xsl:if  test=”$color”> ….</sxl:if>
```

## Elemento <xsl:apply-templates>

Cuando se aplica una plantilla a un nodo, el nodo y todos sus descendientes se sustituyen por el resultado de la aplicación de la plantilla, lo que nos haría perder a los descendientes. Si antes de aplicar la plantilla a un nodo se quiere aplicar a los descendientes las plantillas que les correspondan, hay que utilizar la instrucción <xsl:apply-templates />

Puede tener un atributo **select** que es un XPath.  Este elemento le dice al procesador que busque y aplique cualquier plantilla que coincida con el atributo select.

Un nodo sólo puede tener un template.
Para aplicar un sort en un apply-templates hay que poner el atributo select.
Ejemplo:

```
<?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>My CD Collection</h2>
 <xsl:apply-templates/>
 </body>
 </html>
</xsl:template>

<xsl:template match="cd">
 <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
 </p>
</xsl:template>

<xsl:template match="title">
 Title: <span style="color:#ff0000">
 <xsl:value-of select="."/></span>
 <br />
</xsl:template>

<xsl:template match="artist">
 Artist: <span style="color:#00ff00">
 <xsl:value-of select="."/></span>
 <br />
</xsl:template>

</xsl:stylesheet>
```

Bibliografia:

www.w3schools.com
http://www.mclibre.org/consultar/xml/lecciones/xml_xslt.html