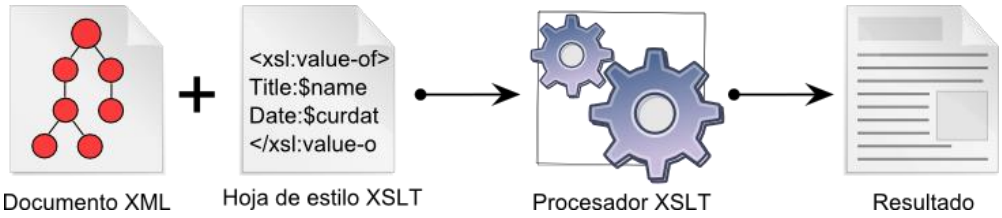
El lenguaje de programación XSLT

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.
- 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.

El lenguaje XSLT está normalizado por el W3C que ha publicado dos versiones de este lenguaje:

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

Aunque hay incompatibilidades entre estas dos versiones, lo que se cuenta en esta manual es válido para ambas versiones.

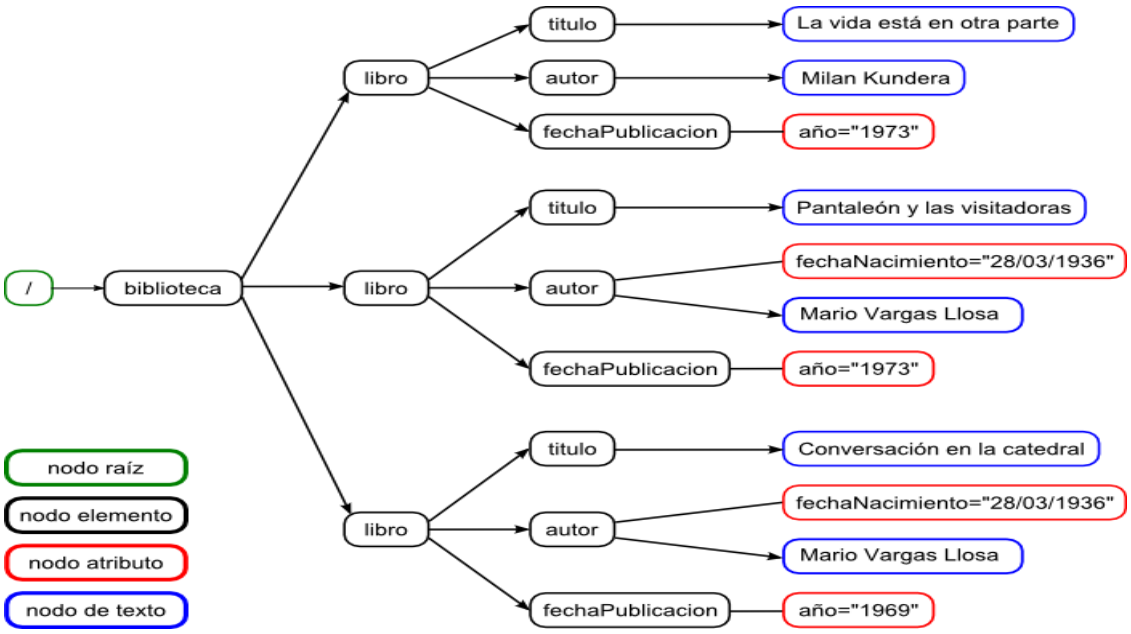
Hojas de estilo XSLT

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:

A) El procesador analiza el documento y construye el árbol del documento.
Dado el siguiente documento XML, a continuación se muestra el árbol que se generaría.

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor  fechaNacimiento="28/03/1936">Mario  Vargas
Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor  fechaNacimiento="28/03/1936">Mario  Vargas
Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```



B) El procesador va recorriendo todos los nodos desde el nodo raíz, aplicando a cada nodo una plantilla, sustituyendo el nodo por el resultado.

C) Cuando el procesador 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, 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, es decir, eliminará el nodo, aunque conservará el texto contenido en el elemento).

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 />**, 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>
```

Enlazar documentos XML con hojas de estilo XSLT

Se puede asociar de forma permanente una hoja de estilo XSLT a un documento XML mediante la instrucción de procesamiento **<?xml-stylesheet ?>**, la misma que permite asociar hojas de estilo CSS. La instrucción de procesamiento **<?xml-stylesheet ... ?>** va al principio del documento, después de la declaración XML.

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

Cuando se visualiza en un navegador web un documento XML enlazado con una hoja de estilo XSLT, los navegadores muestran el resultado de la transformación, aunque si se muestra el código fuente de la página, los navegadores muestran el documento XML original.

Nota: Google Chrome no muestra los documentos xml que enlazan a hojas de estilo XSLT abiertos como archivos locales (file://...), como se

comenta en la lección de diferencias entre navegadores. Firefox e Internet Explorer sí lo hacen.

Ejemplos de plantillas XSLT

Vamos a ver ejemplos de plantillas trabajando sobre el documento siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor      fechaNacimiento="28/03/1936">Mario      Vargas
Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor      fechaNacimiento="28/03/1936">Mario      Vargas
Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

Plantillas vacías o no existentes

- Si no hay plantillas, el procesador simplemente extrae el texto contenido por los nodos.

  En el ejemplo siguiente, el resultado incluye el contenido de los nodos <título> y <autor> puesto que no hay ninguna plantilla.

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

</xsl:stylesheet>
```

- Si hay una plantilla vacía, el procesador sustituye el nodo y todos sus subelementos por nada y no extrae el texto contenido por ese nodo o sus subelementos.

  En el ejemplo siguiente, el resultado incluye el contenido de los nodos <titulo>, ya que no hay regla para ellos, pero los de <autor> se pierden porque la plantilla es vacía.

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

        <xsl:template match="autor">
        </xsl:template>

</xsl:stylesheet>
```

- En el caso más extremo, si la plantilla vacía se aplica al nodo raíz, el procesador sustituye el nodo raíz y todos sus descendientes por nada y no extrae el texto contenido en ningún subelemento.

  En el ejemplo siguiente, el resultado no incluye nada porque la única plantilla ha sustituido el nodo raíz y todos sus subelementos.

```
<?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:template>

</xsl:stylesheet>
```

La instrucción `<xsl:value-of>`

La instrucción **`<xsl:value-of>`** extrae el contenido del nodo seleccionado.

- En el ejemplo siguiente, los títulos de los libros se han perdido, porque el nodo <libro> (y sus subnodos <autor> y <titulo>) se sustituye por el contenido del nodo <autor>.

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

  <xsl:template match="libro">
     <xsl:value-of select="autor"/>
  </xsl:template>


</xsl:stylesheet>
```

- En el ejemplo siguiente, se obtienen el titulo y el autor de los libros.

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

  <xsl:template match="libro">
     <xsl:value-of select="titulo"/>
     <xsl:value-of select="autor"/>
  </xsl:template>


</xsl:stylesheet>
```

- En el ejemplo siguiente, los autores se obtienen gracias a la regla que extrae el contenido del nodo (el caracter punto "." hace referencia al propio elemento) y los títulos se obtienen porque al no haber reglas para ese nodo se extrae el contenido. El resultado es el mismo que el del ejemplo 1-1, pero por motivos distintos.

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

  <xsl:template match="autor">
     <xsl:value-of select="."/>
  </xsl:template>


</xsl:stylesheet>
```

También se pueden extraer los valores de los atributos, utilizando @.

- En el ejemplo siguiente, las fechas de publicación se obtienen gracias a la regla que extraen el valor del atributo y los títulos y autores se obtienen porque al no haber reglas para ese nodo se extrae el contenido.

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

  <xsl:template match="fechaPublicacion">
     <xsl:value-of select="@año"/>
  </xsl:template>


</xsl:stylesheet>
```

Generar texto adicional

Se puede generar texto escribiéndolo en la regla, por ejemplo, código html.

- En el ejemplo siguiente se obtienen los nombres de los autores porque la regla selecciona el nodo <libro> como en el ejemplo, 2-1, pero además generamos las etiquetas <p>. El resultado sigue sin verse bien en el navegador, porque aunque hay etiquetas <p>, falta la etiqueta global <html>.

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

  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
  </xsl:template>


</xsl:stylesheet>
```

Dentro de la regla podemos hacer referencia a varios subnodos.

- En el ejemplo siguiente se obtienen los nombres de los autores y los títulos de los libros porque la regla selecciona el nodo <libro> como en el ejemplo, 2-1, pero además generamos las etiquetas <p>. El resultado dependiendo del navegador podrá o no verse, porque aunque hay etiquetas <p>, falta la etiqueta global <html>.

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

  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
     <p><xsl:value-of select="titulo"/></p>
  </xsl:template>


</xsl:stylesheet>
```

- Los siguientes ejemplos intentan conseguir el mismo resultado que el ejemplo anterior (ejemplo 3-2), pero utilizando dos reglas, y no lo consiguen:

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

  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
 </xsl:template>

  <xsl:template match="libro">
     <p><xsl:value-of select="titulo"/></p>
  </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="libro">
     <p><xsl:value-of select="titulo"/></p>
  </xsl:template>
```

```
  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
  </xsl:template>
</xsl:stylesheet>
```

- ¿Por qué en un caso se obtienen sólo los títulos y en el otro sólo los autores? Porque el procesador XSLT sólo aplica una regla a cada nodo. Si tenemos dos reglas para el mismo nodo, el procesador sólo aplica una de ellas (la última, en este caso).

Además de generar etiquetas, se puede generar texto.

- En el ejemplo siguiente se generan frases a partir del contenido de los nodos.

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

     <xsl:template match="libro">
          <p><xsl:value-of select="autor"/>
          escribió "<xsl:value-of
          select="titulo"/>"</p>
     </xsl:template>


</xsl:stylesheet>
```

Aplicar reglas a subnodos: la instrucción `<xsl:apply-templates>`

La instrucción **`<xsl:apply-templates>`** hace que se apliquen a los subelementos las reglas que les sean aplicables.

- En el ejemplo siguiente, se genera la etiqueta *<html>* además de unos párrafos con los nombres de los autores. Este ejemplo dependiendo del navegador se podrían ahorrar las etiquetas <html> y </html>.

```
<?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>
      <h1>Autores</h1>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
  </xsl:template>


</xsl:stylesheet>
```

- La primera regla sustituye el elemento raíz (y todos sus subelementos) por las etiquetas *<html>* y *<h1>*, pero además aplica a los subelementos las reglas que les son aplicables. En este caso, sólo hay una regla para los elementos <libro> que generan los párrafos.

La instrucción ***<xsl:apply-templates select="PATRÓN">*** hace que se apliquen (en ese momento) a los subelementos indicados en ***PATRÓN*** las reglas que les sean aplicables.

- En el siguiente ejemplo, dentro de la etiqueta libro se crea un párrafo y se aplica la plantilla al título. Esta plantilla consiste en extraer el título y escribirlo en color azul.

```
<?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="libro">
     <p style="color:#00C"><xsl:apply-templates
select="titulo"/></p>
  </xsl:template>
```

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


</xsl:stylesheet>
```

La instrucción `<xsl:text>`

La instrucción **`<xsl:text>`** permite generar texto que no puede generarse simplemente añadiéndolo como en los ejemplos anteriores.

- En el ejemplo siguiente ponemos entre "< … >" la palabra autores.

```
<?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>
      <h1><xsl:text>&lt;  </xsl:text>
Autores<xsl:text>&gt;  </xsl:text>
</h1>
      <xsl:apply-templates />
    </html>
  </xsl:template>

  <xsl:template match="libro">
     <p><xsl:value-of select="autor"/></p>
  </xsl:template>


</xsl:stylesheet>
```

La instrucción `<xsl:attribute>`

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>
```

- No se puede utilizar la instrucción **`<xsl:value-of>`** como en el ejemplo incorrecto 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="licencia">
    <p><img src="<xsl:value-of select="imagen"
/>" /></p>
  </xsl:template>


</xsl:stylesheet>
```

- En este caso el problema no es debido a la utilización de comillas dobles (también daría error si se hubieran utilizado comillas simples o entidades), sino que es necesario utilizar la instrucción **`<xsl:attribute>`**.
- Para generar un atributo en una etiqueta, es necesario utilizar la instrucción **`<xsl:attribute>`**, como en el ejemplo siguiente:

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

- En la hoja de estilo XSLT, la etiqueta *<img>* se escribe con apertura y cierre, aunque en la realidad la etiqueta *<img>* monoatómica.
- Como en ejemplos anteriores, para que la imagen se muestre en algunos navegadores sería necesario generar también la etiqueta *<html>*:

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

</xsl:stylesheet>
```

La instrucción <xsl:for-each>

La instrucción <xsl:for-each> aplica una acción repetidamente para cada nodo de un conjunto.

- En el siguiente ejemplo, para cada etiqueta libro que encuentre en el fichero mostrará su título y autor.

```
<?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>
       <h2>Informe detallado</h2>
     <xsl:for-each select="biblioteca/libro">
       <strong>Título:</strong><em><xsl:value-of
select="titulo" /></em><br/>
       <strong>Autor:</strong><em><xsl:value-of
select="autor" /></em><br/><hr/>
     </xsl:for-each>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

- En el siguiente ejemplo, para cada etiqueta libro del autor "*Mario Vargas Llosa*" que encuentre en el fichero mostrará su título y autor.

```
<?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>
       <h2>Informe detallado</h2>
     <xsl:for-each
select="biblioteca/libro[autor='Mario Vargas
Llosa']">
       <strong>Título:</strong><em><xsl:value-of
select="titulo" /></em><br/>
       <strong>Autor:</strong><em><xsl:value-of
select="autor" /></em><br/><hr/>
     </xsl:for-each>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

La instrucción `<xsl:if>`

La instrucción `<xsl:if>` sirve para evaluar condiciones sobre valores de atributos o elementos. Si la comparación es cierta se evalúa el contenido del elemento "<xsl:if>", en caso contrario no se evalúa.

- Los operadores lógicos que se pueden utilizar para cambiar el patrón de búsqueda o filtro son los siguientes:
  - Operador de igualdad: =
  - Operador de desigualdad: !=
  - Operador menor que: <
  - Operador mayor o igual que: >=
  - Operador menor o igual que: <=
  - Operador mayor que: >

- El siguiente ejemplo nos muestra una lista de los libros publicados después de 1970:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="biblioteca">
    <html>
       <h2>Libros publicados después de 1970</h2>
       <ul>
       <xsl:apply-templates select="libro"/>
       </ul>
    </html>
  </xsl:template>
  <xsl:template match="libro">
       <xsl:if test="fechaPublicacion[@año &gt;
    1970]">

          <li>Título:<xsl:value-of select="titulo" />
    , Autor:<xsl:value-of select="autor" /></li>
       </xsl:if>
  </xsl:template>

</xsl:stylesheet>
```

La instrucción `<xsl:choose>`

La instrucción `<xs:chose>` sirve para evaluar condiciones múltiples. Equivaldría a la instrucción *switch* de algunos lenguajes de programación

- La sintaxis sería la siguiente:

```
<xsl:choose>
      <xsl:when test="EXPRESIÓN 1"> ... </xsl:when>
      <xsl:when test="EXPRESIÓN 2"> ... </xsl:when>
      .....
     <xsl:otherwise> ... </xsl:otherwise>
</xsl:choose>
```

- El siguiente ejemplo nos muestra en una lista en color rojo los libros publicados antes de 1970 y en color azul los publicados después de 1970:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="biblioteca">
    <html>
      <h2 style="color:red">Libros publicados antes
de 1970</h2>
      <h2 style="color:blue">Libros publicados
después de 1970</h2>
      <ul>
      <xsl:apply-templates select="libro"/>
      </ul>
    </html>
  </xsl:template>
  <xsl:template match="libro">
      <xsl:choose>
            <xsl:when test="fechaPublicacion[@año
&lt; 1970]">

     <li style="color:red">Título:<xsl:value-of
select="titulo" /> , Autor:<xsl:value-of
select="autor" /></li>
            </xsl:when>
```

```
            <xsl:when test="fechaPublicacion[@año
&gt; 1970]">

     <li style="color:blue">Título:<xsl:value-of
select="titulo" /> , Autor:<xsl:value-of
select="autor" /></li>
            </xsl:when>
       </xsl:choose>
  </xsl:template>

</xsl:stylesheet>
```

La instrucción `<xsl:sort>`

La instrucción `<xs:sort>` ordena un conjunto de nodo de acuerdo a algún elemento.

- La sintaxis sería la siguiente:

  <xsl:sort select="PATRÓN" order="ascending" />

  el orden puede ser ***"ascending*** o ***descending***", y por defecto es "***ascending".***

- El siguiente ejemplo nos muestra en una lista los libros ordenados por títulos:

```
<?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>
      <h2>Libros ordenados por título</h2>
      <ul>
     <xsl:for-each select="biblioteca/libro">
      <xsl:sort select="titulo" order="ascending" />
      <li><xsl:value-of select="titulo" /></li>
     </xsl:for-each>
      </ul>
    </html>
  </xsl:template>

</xsl:stylesheet>
```