Tema 12

Transformación de información. XSLT

Objetivos

- Justificar la necesidad de la transformación o conversión de documentos XML e identificar su ámbito de aplicación.
- Analizar las tecnologías implicadas en la transformación de la información sobre documentos XML y su modo de funcionamiento.
- Describir la estructura y sintaxis específica utilizada en la conversión y adaptación de documentos XML.
- Crear especificaciones de transformación de información sobre documentos XML.
- Identificar y caracterizar herramientas específicas relacionadas con la conversión de documentos XML.

Introducción

En los dos temas anteriores se ha abordado la capacidad de los documentos XML para almacenar información estructurada en formato XML, así como las técnicas existentes para poder acceder a la información almacenada. En este tema se aborda el estudio de las técnicas que permiten transformar o convertir un documento XML en otro documento de un tipo distinto como: HTML, PDF, texto, RTF, etc. De esta manera, el documento resultante de la transformación puede presentar visualmente la información almacenada en el documento XML inicial de una manera adecuada. El contenido más importante de este tema se refiere a la aplicación de la tecnología XSLT *(eXtensible Stylesheet Language Transformations)*. Las hojas de transformaciones XSLT se utilizan juntamente con el lenguaje de consulta XPath que se aplica para buscar y seleccionar la información que se desea incorporar en el documento resultante.

Índice

12.1 VISUALIZACIÓN DE DOCUMENTOS XML

Un documento XML no dispone de una visualización concreta sobre un navegador, porque su contenido no define ningún aspecto de visualización. Efectivamente, los documentos XML simplemente almacenan información en formato texto. Como ya se ha podido comprobar en los temas anteriores, cuando se abre un documento XML sobre un navegador se muestra su árbol de nodos, si es que el documento XML está bien formado.

Existen diferentes maneras de presentar visualmente los datos almacenados en un documento XML:

- **Utilizando una hoja de estilo CSS**. Se construye una hoja de estilo CSS específica que indica al navegador cómo debe presentar la información almacenada en cada elemento del documento XML. De esta manera se produce una conversión de cada elemento del documento XML en un elemento visual de HTML.

- **Utilizando una hoja de transformaciones XSL**. Se construye una hoja de transformaciones XSL para definir la forma cómo el procesador XSL convertirá un documento XML en otro documento de otro tipo como, por ejemplo: HTML, PDF, RTF, etc. Esta conversión se realiza atendiendo al formateo de la salida que se define en la hoja de transformaciones XSL.

- **Utilizando un Lenguaje de programación**. Se emplea un lenguaje de programación procedural, como Java, C, Javascript, Basic, etc. para construir un programa que procese el documento XML y obtenga el resultado de presentación visual deseado.

En este tema se estudian los dos primeros métodos. El uso de hojas de estilo se estudia brevemente debido a sus limitaciones. Se incide, especialmente, en la creación y especificación de las hojas de transformaciones XSL, mediante la utilización del lenguaje estándar XSLT, como método más adecuado para realizar conversiones de información sobre los documentos XML. El tercer método excede de los objetivos de este curso, aunque puede tener su interés en aplicaciones concretas.

12.2 APLICACIÓN DE CSS EN DOCUMENTOS XML

Una hoja de estilo CSS consiste en un conjunto de reglas de estilo con prioridad que se aplican a los distintos elementos o etiquetas que forman un documento HTML. Sin embargo, las hojas de estilo CSS también pueden aplicarse a un documento XML.

En el lenguaje de marcas HTML las etiquetas están definidas y tienen una función establecida. Por lo tanto, los navegadores, u otros programas de procesamiento, presentan los elementos HTML de una manera predeterminada mediante un estilo inicial que puede adaptarse a las características de visualización deseadas mediante hojas de estilo específicas. En los documentos XML, las etiquetas que se pueden definir son infinitas y su función depende de cada caso concreto. Por lo tanto, los navegadores, u otros programas de procesamiento, no pueden tener información predeterminada para la presentación de los documentos XML. De manera que, la presentación visual de los documentos XML depende exclusivamente de la información de estilo que se pueda incorporar.

Un ejemplo sencillo

A continuación, se analizará un ejemplo de uso sencillo para comprender el mecanismo de aplicación de una hoja de estilo CSS a un documento XML. Sea el siguiente documento XML:

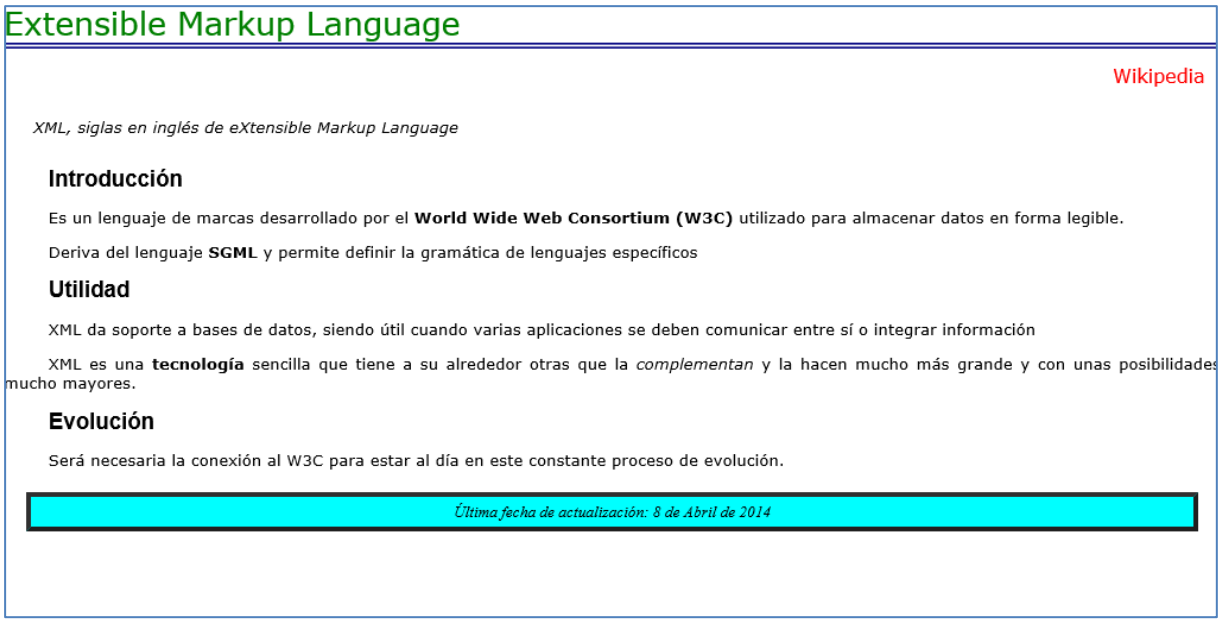
```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet href="EjemploCSS_1.css" type="text/css"?>
<articulo>

  <!-- Cabecera del documento -->
  <tituloP>
    Extensible Markup Language
  </tituloP>
  <autor>
    Wikipedia
  </autor>
  <intro>
    XML, siglas en inglés de eXtensible Markup Language
  </intro>

  <!-- Cuerpo del documento -->
  <tituloA>
    Introducción
  </tituloA>
  <p>
    Es un lenguaje de marcas desarrollado por el <n>World Wide
    Web Consortium (W3C)</n> utilizado para almacenar datos en
    forma legible.
  </p>
  <p>
    Deriva del lenguaje <n>SGML</n> y permite definir la gramática
    de lenguajes específicos
  </p>
  <tituloA>Utilidad</tituloA>
  <p> XML da soporte a bases de datos, siendo útil cuando varias
    aplicaciones se deben comunicar entre sí o integrar información </p>
  <p> XML es una <n>tecnología</n> sencilla que tiene a su alrededor
    otras que la <c>complementan</c> y la hacen mucho más grande y
    con unas posibilidades mucho mayores. </p>
  <tituloA>Evolución</tituloA>
  <p>Será necesaria la conexión al W3C para estar al día en este
    constante proceso de evolución.</p>

  <!-- Comentarios del documento -->
  <comentarioF>
    Última fecha de actualización: 8 de Abril de 2014
  </comentarioF>
  <comentario_oculto>
    Este documento es un extracto
  </comentario_oculto>
</articulo>
```

En el código anterior, puede apreciarse la instrucción de procesamiento de tipo <?xml-stylesheet> que está situada en el prólogo del documento. Esta instrucción permite relacionar el documento XML con el archivo externo de hoja de estilo CSS. El contenido del archivo de hoja de estilo externo, denominado EjemploCSS_1.css, es el siguiente:

```
tituloP { display:block; width:100%; font-size:20pt; font-family:Verdana; color:green;
          border-bottom:double 4pt navy; text-align:left }
autor { display:block; font-size:12pt; font-family:Verdana; color:red;
        text-align:right; margin:10pt }
intro { display:block; margin:25px; font-family:Verdana; font-size:10pt;
        font-style:italic }
tituloA { display:block; font-family:Arial; margin-left:1cm; text-align:left;
          font-size:14pt; font-weight:bold; margin-bottom:10pt }
comentarioF { display:block; margin:.2in; padding:5px; font-style:italic;
              border:outset 3pt; text-align:center; font-size:10pt; background-color:aqua }
p { display:block; margin-bottom:10pt; font-family:Verdana,sans-serif; font-size:10pt;
    text-align:justify; text-indent:1cm }
n { font-weight:bold; display:inline }
c { font-style:italic; display:inline }
comentario_oculto { display:none }
```

En el código anterior, puede observarse cómo se definen las propiedades de estilo asociadas a cada uno de los elementos del documento XML, para así dar el formato de visualización deseado a cada elemento. Debe tenerse en cuenta que, de manera predeterminada, todos los elementos XML son elementos de línea. Y, por tanto, es necesario incluir la propiedad display, de modo que se puedan definir los elementos de bloque (block) o de línea (inline) de manera adecuada.

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador.



Otro ejemplo sencillo

También es posible incluir etiquetas del lenguaje HTML dentro de los documentos XML para tratar de ayudar a especificar la presentación visual del documento. Para ello, se incluye la definición del espacio de nombres correspondiente de HTML, de manera que se pueden diferenciar las etiquetas específicas del lenguaje HTML de aquellas otras etiquetas que pudieran tener el mismo nombre en el documento XML. Sea el siguiente documento XML:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="EjemploCSS_2.css" type="text/css"?>
<articulo xmlns:html="http://www.w3.org/1999/xhtml">
  <!-- Cabecera del documento -->
  <html:a name="principio"></html:a>
  <tituloP>Extensible Markup Language</tituloP>
  <autor> <html:a href="mailto:fedrico@sanchez.org">Federico Sánchez</html:a> </autor>
  <intro>XML, siglas en inglés de eXtensible Markup Language</intro>
  <!-- Cuerpo del documento -->
  <tituloA>Introducción</tituloA>
  <p> Desarrollado por el <n>World Wide Web Consortium (W3C)</n>  Deriva del lenguaje
      <n>SGML </n> y permite definir la gramática de lenguajes específicos </p>
  <tituloA>Utilidad</tituloA>
  <p> XML es una <n> tecnología </n> sencilla que tiene a su alrededor otras que la
      <c> complementan </c> y la hacen mucho más grande y con unas posibilidades mucho
      mayores. </p>
  <html:br />
  <tituloA>Evolución</tituloA>
  <p> Será necesaria la conexión al W3C para estar al día en este constante proceso de
      evolución.</p>
  <html:br />
  <tituloA>Aclaración</tituloA>
  <p>Las normas necesarias para el desarrollo de XML van publicándose poco a poco, siendo
     necesaria la conexión al <html:a href="http://www.w3.org">W3C</html:a> para estar al
     día en este constante proceso de evolución.</p>
  <!-- Comentarios del documento -->
  <comentarioF>Última fecha de actualización: 8 de Abril de 2014</comentarioF>
  <html:p> Esto es un párrafo de html</html:p>
  <p>
  <html:center>
  <html:a href="#principio">[Principio de página] </html:a>
  <html:font color="white">----</html:font>
  <html:a href="http://es.wikipedia.org/wiki/XML"> [Página principal]</html:a>
  </html:center>
  </p>
  <comentarioOculto>Este documento es un extracto</comentarioOculto>
</articulo>
```

En el código anterior puede apreciarse que se ha incluido la definición del espacio de nombres de HTML. Y puede observarse que, por ejemplo, la etiqueta <html:p> define un párrafo de HTML, mientras que la etiqueta <p> hace referencia a un elemento del documento XML.

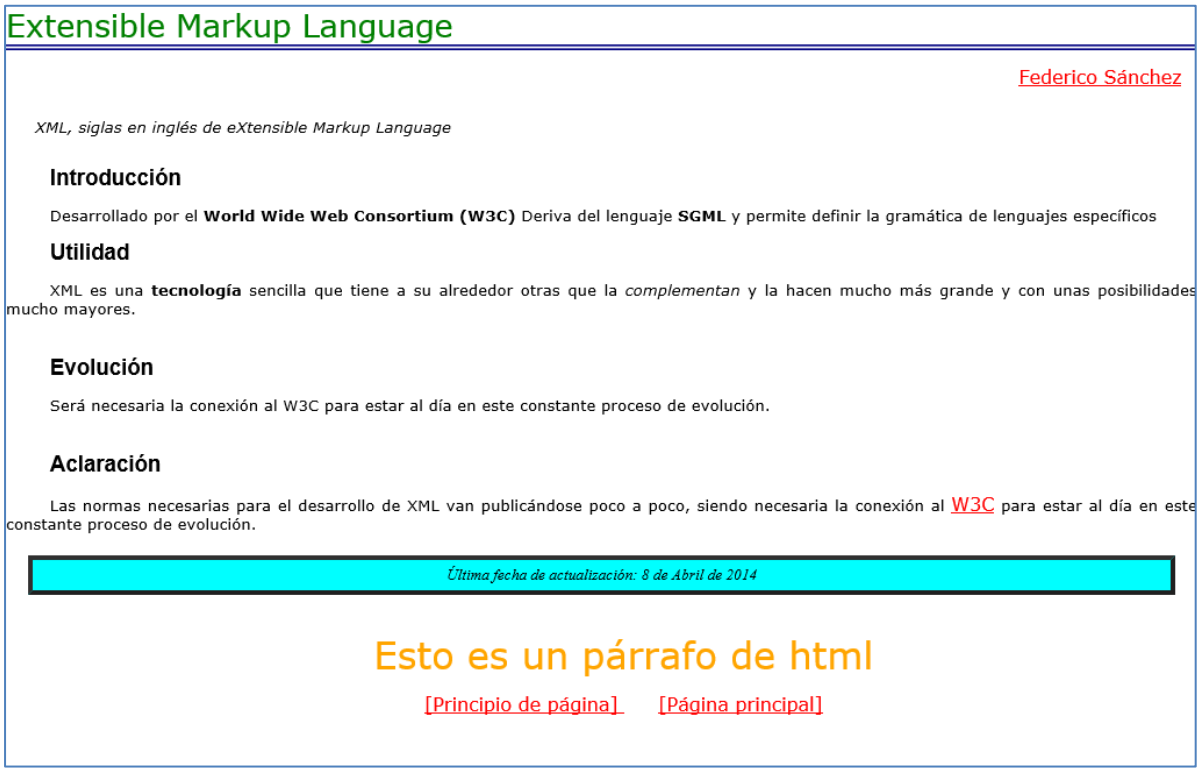
El contenido del archivo de hoja de estilo CSS, denominado EjemploCSS_2.css, es el siguiente:

```
@namespace html "http://www.w3.org/1999/xhtml";
tituloP { display:block; width:100%; font-size:20pt; font-family:Verdana;
          color:green; border-bottom:double 4pt navy; text-align:left }
autor { display:block; font-size:12pt; font-family:Verdana; color:red;
        text-align:right; margin:10pt }
intro { display:block; margin:25px; font-family:Verdana; font-size:10pt;
        font-style:italic }
tituloA { display:block; font-family:Arial; margin-left:1cm; text-align:left;
          font-size:14pt; font-weight:bold; margin-bottom:10pt }
comentarioF { display:block; margin:.2in; padding:5px; font-style:italic;
              border:outset 3pt; text-align:center; font-size:10pt;
              background-color:aqua }
p { display:block; margin-bottom:10pt; font-family:Verdana,sans-serif;
    font-size:10pt; text-align:justify; text-indent:1cm }
```

```
n { font-weight:bold; display:inline }
c { font-style:italic; display:inline }
comentarioOculto {display:none}
html|a { text-decoration:underline; color:red; font-family:Verdana;
         font-size:12pt }
html|p { color:orange; font-family:Verdana; font-size:24pt;
         text-align:center }
```

En el código anterior, puede observarse que también se ha definido el espacio de nombres de HTML en el archivo de hoja de estilo externa. También puede apreciarse que, para dar estilo a un elemento de HTML se utiliza el conector "|" para unir el prefijo y el nombre del elemento correspondiente.

El ejemplo anterior, produce el siguiente resultado en la ventana de visualización del navegador:



La utilización de hojas de estilo CSS como mecanismo para definir la presentación visual de la información contenida en un documento XML tiene grandes limitaciones. Solo es aconsejable si el contenido del documento está formado solo por texto y si los datos se estructuran de una forma sencilla y secuencial. Además, no es posible seleccionar la información y, consecuentemente, se presentará siempre toda la información contenida en el documento XML.

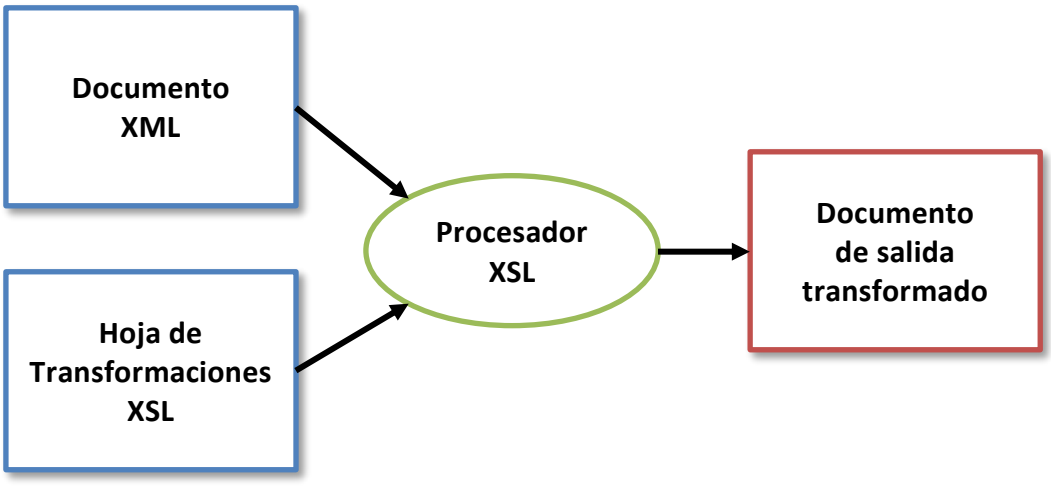
El **método recomendado para definir la presentación visual** de la información almacenada en documentos XML **consiste en aplicar transformaciones XSL sobre los documentos XML**. En los siguientes apartados se estudia la familita de lenguajes XSL y el lenguaje estándar XSLT. El lenguaje XSLT es una tecnología específica para realizar la conversión y adaptación visual de documentos XML que amplía las capacidades de presentación visual de la información en formato XML.

12.3 LA FAMILIA DE LENGUAJES XSL

La familia de lenguajes XSL *(eXtensible Stylesheet Language)*, o lenguajes extensibles de hoja de estilo, es una familia de lenguajes basados en el estándar XML que permiten expresar cómo la información contenida en un documento XML debe ser transformada, o formateada, para conseguir una presentación visual adecuada en un determinado medio de información. XSL es, por tanto, una familia de lenguajes que permiten expresar hojas de estilo para documentos XML.

Algunos autores sostienen que XSL es para XML, lo que CSS es para HTML. Por este motivo, los documentos XSL suelen denominarse hojas de estilo de XML. Sin embargo, no debe confundirse CSS con XSL, ni quiere decir que uno de ellos vaya a sustituir al otro, aunque ambos tengan la función de facilitar la presentación de información en un formato visual más legible y atractivo. Desde el inicio del uso del lenguaje XML ha habido un cierto interés por proporcionar un mecanismo que facilite la presentación visual de los documentos XML en la Web. Por este motivo se adaptaron algunas de las tecnologías existentes en ese momento, como por ejemplo CSS. No obstante, en el apartado anterior se han podido comprobar las importantes limitaciones del uso del lenguaje de estilo CSS en este ámbito. Estas limitaciones, que están marcadas por la falta de construcciones de sentencias de control y de filtros de información adecuados, hacen que el uso conjunto de CSS con XML sea insuficiente, especialmente si se compara con las ventajas que proporciona la aplicación de las técnicas de transformación de documentos que proporciona XSL. Se puede concluir que la familia de lenguajes XSL permite definir hojas de estilo adecuadas para la presentación visual de los documentos XML, mientras que CSS utiliza un método de presentación que está adaptado específicamente al lenguaje HTML. En la práctica, ambos lenguajes, XSL y CSS, son complementarios, de modo que XSL se suele utilizar con documentos XML y CSS con documento HTML.

La familia de lenguajes XSL se utilizan para extraer información de documentos XML y generar otros documentos con formatos de salida diferentes, tales como: XML, HTML, PDF, texto plano, RTF, PostScript, etc. El proceso habitual consiste en convertir los datos contenidos en un documento XML en otro documento con un formato de salida adecuado, a partir de una hoja de transformación, siguiendo un mecanismo de transformación como el que se muestra en la siguiente ilustración.



La familia de lenguajes extensibles de hoja de estilo, XSL, son un conjunto de recomendaciones que definen la transformación y presentación de documentos XML. Consta de tres partes o lenguajes:

- **XSLT *(eXtensible Stylesheet Language Transformations)* o Transformaciones XSL**. Es un lenguaje que se utiliza para transformar documentos XML en formato HTML o XML.

- **XPath *(XML Path Language)***. Es un lenguaje de expresiones utilizado por XSLT para referenciar o acceder a partes de documentos XML. Se estudió en un tema anterior.

- **XSL-FO (*eXtensible Stylesheet Language Transformations Formatting Objects)***. Es un lenguaje de formato para documentos XML. Permite generar documentos en formatos PDF, RTF, PostScript, etc. a partir de documentos XML. Cabe destacar que el documento XSL-FO puede contener tanto los datos a presentar como la descripción del formato a aplicar.

Utilizando los tres componentes anteriores, se puede:

- Transformar un documento XML en otro documento diferente de otro tipo de formato.

- Filtrar y ordenar la información contenida en un documento XML.

- Definir partes en un documento XML.

- Formatear un documento XML sobre la base de los valores de la información almacenada.

- Extraer datos contenidos en un documento XML para aplicar una transformación XSL-FO para generar archivos PDF.

La familia de lenguajes XSL fue estandarizada por el organismo W3C en varias etapas: la primera, en noviembre de 1999, incluía XSLT y XPath; la segunda, en octubre del 2000, incluía la recomendación XSL-FO; posteriormente, se han publicado nuevas versiones de cada uno de los lenguajes. Para obtener más información, se puede visitar la dirección web: http://www.w3.org/Style/XSL.

Este tema se centra, principalmente, en el estudio del lenguaje XSLT y en la creación de hojas de transformaciones XSLT, también denominadas hojas de estilo XSLT, que permiten generar documentos en formatos de salida XML, HTML o texto a partir de documentos XML. El lenguaje XSLT utiliza el lenguaje XPath para seleccionar la información contenida en el documento XML origen.

12.4 EL LENGUAJE XSLT

El lenguaje XSLT es la especificación concreta que, dentro de la familia de lenguajes XSL, sirve para definir el proceso de transformación de un documento XML en otro documento de destino, principalmente, en formato XML o HTML. Una vez aclaradas las diferencias entre XSL y XSLT, hay que tener en cuenta que, habitualmente, ambos términos se usan en la práctica como equivalentes para referirse al proceso de transformación de documentos XML.

La transformación XSLT se realiza convirtiendo cada elemento de un documento XML en otro elemento de salida diferente. El lenguaje XSLT permite añadir o eliminar elementos a la salida, reordenar o recolocar elementos, evaluar condiciones, seleccionar qué elementos del documento XML origen se desea mostrar, etc. durante el proceso de transformación.

El lenguaje XSLT utiliza el lenguaje XPath para especificar o referenciar las partes del documento XML que cumplen uno o más patrones predefinidos. Cuando se encuentra una coincidencia con los patrones predefinidos, el procesador de XSLT transformará la parte coincidente del documento XML origen sobre el documento destino. Las partes que no son coincidentes no se transforman y, consecuentemente, quedarán en el documento destino sin ningún cambio. Aunque el lenguaje XPath fue diseñado, en su origen, para ser utilizado de forma independiente, tiene su utilidad más completa cuanto aparece embebido en una hoja de transformación XSLT, de modo que permite crear filtros de información que permiten obtener los elementos a traspasar al documento de salida.

Un primer ejemplo de transformación XSLT de un documento XML en un documento HTML

A continuación, se incluye un ejemplo de transformación XSLT de un documento XML, denominado *EjemploXSLT_1.xml*, en un documento HTML de destino. Sea el siguiente documento XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet href="EjemploXSLT_1.xsl" type="text/xsl"?>
<strings>
   <s>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
      eiusmod tempor incidunt ut labore et dolore magna aliqua.
   </s>
   <s>
      Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquid ex ea commodi consequat.
   </s>
   <s>
      Quis aute iure reprehenderit in voluptate velit esse cillum
      dolore eu fugiat nulla pariatur. Excepteur sint obcaecat
      cupiditat non proident, sunt in culpa qui officia deserunt
      mollit anim id est laborum.
   </s>
</strings>
```

En el código anterior puede observarse cómo en el prólogo del documento XML se ha incluido una instrucción de procesamiento de tipo <?xml-stylesheet href=”...” type=”text/xsl”?>. Esta instrucción especifica el acceso al archivo que define la hoja de transformaciones XSLT asociada al documento XML. El acceso a este archivo asociado, que tiene extensión .xsl, se especifica mediante el atributo href, cuyo valor debe incluir el nombre del archivo, así como la ruta relativa desde la ubicación actual del archivo correspondiente al documento XML a transformar.

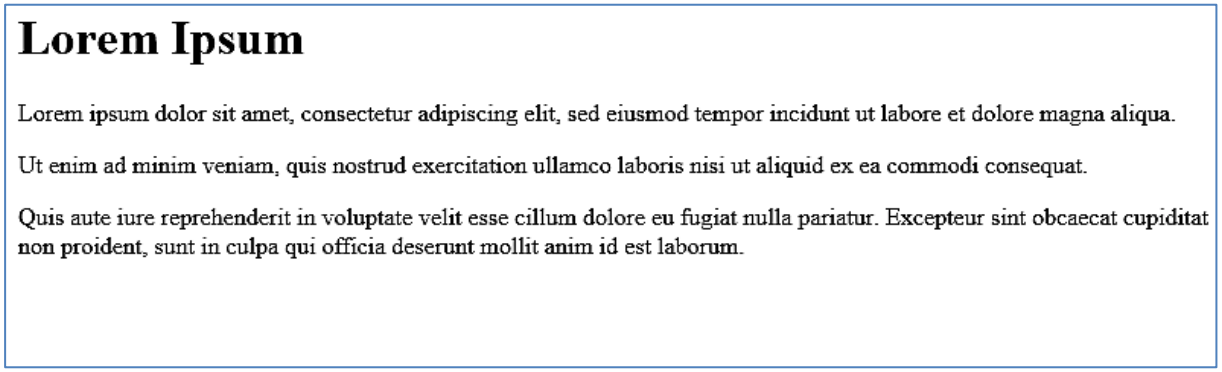
El siguiente código, que se corresponde con la hoja de transformaciones XSLT denominada *EjemploXSLT_1.xsl*, transforma el documento XML anterior en un documento HTML de salida. En este ejemplo, el archivo de transformaciones, *EjemploXSLT_1.xsl*, y el archivo del documento XML, *EjemploXSLT_1.xml*, deberán estar situados en la misma carpeta.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Ejemplo 1 de XSLT</title>
      </head>
      <body>
        <h1>Lorem Ipsum</h1>
         <xsl:for-each select = "/strings/s">
           <p>
             <xsl:value-of select="." />
           </p>
         </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Considerando el código anterior, el procesador o motor XSL realizará las siguientes acciones:

- Busca la raíz del documento XML en el árbol de nodos del documento XML origen. Esta búsqueda corresponde con el código del lenguaje XSLT: <xsl:template match="/">.
- Compara la raíz del documento XML con el contenido de la plantilla en la hoja de estilo.
- Genera la salida de los elementos HTML sobre el documento de salida.
- Procesa los elementos XSLT de la plantilla. El código <xsl:for-each select = "/strings/s"> hace que se seleccionen todos los elementos <s> que sean hijos del elemento raíz, <strings>. Y, además, hace que para cada uno de los elementos seleccionados se genere un párrafo, elemento <p>, que contiene el valor del elemento, al cual se accede mediante el elemento <xsl:value-of…> de XSLT.

Al abrir el documento XML del ejemplo anterior con un navegador Web moderno que incluya soporte XSLT, se obtiene el siguiente resultado en la ventana de visualización del navegador. El procesador de XSLT incorporado utilizará la hoja de estilos XSLT para realizar la transformación del documento XML de origen en un documento HTML destino, produciéndose el siguiente resultado.



Estructura de una hoja de transformaciones XSLT

En el código del archivo *EjemploXSLT_1.xsl* anterior, puede apreciarse la estructura básica de una hoja de transformaciones XSLT, también denominada hoja de estilos XSLT, que es un archivo de texto compuesto por varias partes:

- Una **declaración de documento XML**: <?xml version="1.0" encoding="utf-8" ?>, puesto que el lenguaje XSLT es un lenguaje basado en el estándar XML.

- Un **elemento raíz de la hoja de transformaciones XSLT** que se define con la etiqueta **<xsl:stylesheet>**, aunque también puede utilizarse la etiqueta **<xsl:transform>**. Ambas etiquetas son equivalentes y válidas. De hecho, suele emplearse indistintamente el término hoja de estilos XSLT, o bien, el término hoja de transformaciones XSLT. Esta etiqueta, además de la declaración el elemento raíz de la hoja de estilos XSLT, identifica la versión de XSL utilizada, así como espacio de nombres recomendado por el W3C para el lenguaje XSL.

- Diversas **porciones de código HTML** que ayudan a formar el nuevo documento HTML de salida o de destino a partir de la transformación del documento XML original.

- Diversos **elementos XSLT** que aparecen como hijos del elemento raíz <xsl:stylesheet>. Estos elementos forman parte del espacio de nombres de prefijo xsl:. Por ejemplo, pueden ser los siguientes elementos XSLT: <xsl:template>, <xs:for-each>, <xsl:value-of>, <xsl:output>, <xsl:sort>, <xsl:if>, <xsl:strip-space>, <xsl:preserve-space>, <xsl:import>, etc.

Elementos XSLT de una hoja de transformaciones XSLT

La siguiente tabla muestra una breve descripción de los elementos XSLT que pueden aparecer en una hoja de transformaciones XSLT.

Elemento XSLT	Descripción
apply-imports	Procesa el nodo actual utilizando solamente reglas de plantilla que han sido importadas.
apply-templates	Busca y aplica la regla de plantilla adecuada al elemento actual o a los nodos secundarios del elemento actual.
attribute	Crea un nodo de atributo y lo añade al elemento resultante.
attribute-set	Define un conjunto de atributos con nombre.
call-template	Invoca a una plantilla con nombre
choose	Se utiliza junto con <xsl:when> y <xsl:otherwise> para expresar pruebas condicionales múltiples.
comment	Crea un nodo de comentario en el árbol de resultados.
copy	Crea una copia del nodo actual (sin atributos ni nodos secundarios).
copy-of	Crea una copia del nodo actual (con atributos y nodos secundarios).
decimal-format	Define los caracteres y símbolos que se utilizarán al convertir números en cadenas, con la función format-number().

Elemento XSLT	Descripción
element	Crea un nodo de elemento en el documento de salida.
fallback	Especifica un código alternativo para ejecutar, si el procesador no admite un elemento XSLT
for-each	Aplica una plantilla repetidamente, aplicándola para cada nodo del conjunto de nodos especificado.
if	Especifica una plantilla que se aplicará sólo si la condición especificada es verdadera.
import	Importa el contenido de una hoja de estilos XSLT externa que está definida en otro archivo. Una hoja de estilos importada tiene una precedencia inferior que la hoja de estilos en la que se importa.
include	Incluye el contenido de una hoja de estilos XSLT externa que está definida en otro archivo. Una hoja de estilo incluida tiene la misma precedencia que la hoja de estilos en la que se incluye.
key	Declara una clave que se puede utilizar con la función key().
message	Escribe un mensaje de texto. Se utiliza para informar de errores.
namespace-alias	Sustituye el prefijo de un espacio de nombres por otro prefijo.
number	Determina la posición entera del nodo actual y formatea un número.
otherwise	Especifica una acción predeterminada para el elemento <xsl:choose>.
output	Define el formato del documento de salida.
param	Declara un parámetro local o global.
preserve-space	Conserva los espacios en blanco en los elementos definidos.
processing-instruction	Escribe una instrucción de procesamiento a la salida.
sort	Especifica los criterios de ordenación de la salida para listas de nodos seleccionadas por <xsl:for-each> o <xsl:apply-tempaltes>.
strip-space	Se eliminan los espacios en blanco en los elementos definidos.
stylesheet	Define el elemento raíz de una hoja de estilo XSLT.
template	Se utiliza para construir plantillas. Una plantilla contiene las reglas de transformación a aplicar sobre el nodo especificado.
text	Escribe texto literal a la salida.
transform	Hace la misma función que <xsl:stylesheet>. Ambos son equivalentes.
value-of	Escribe el valor de un nodo seleccionado en la salida.
variable	Declara una variable local o global.
when	Especifica una acción para el elemento <xsl:choose>.
with-param	Define el valor de un parámetro que se pasa a una plantilla.

A continuación, se estudian los elementos XSLT básicos que suelen utilizarse más habitualmente para realizar transformaciones XSLT sobre documentos XML. Puede encontrarse más información visitando la dirección web: https://msdn.microsoft.com/es-es/library/ms256058(v=vs.120).aspx

Elemento <xsl:template>

El elemento <xsl:template> es uno de los elementos principales y más utilizados del lenguaje de transformaciones XSLT. Como consideración previa, antes de iniciar el estudio este elemento, es necesario comprender que una hoja de estilo XSLT está formada por una serie de **plantillas o *templates* de transformación**.

Cada elemento <xsl:template> representa una plantilla que define una serie de acciones de transformación XSL. Estas acciones se realizarán si el patrón definido en la plantilla coincide con algún elemento o elementos del árbol de nodos del documento XML. Los elementos del documento XML que no son coincidentes con el patrón especificado no se transformarán y, consecuentemente, quedarán en el documento destino sin ningún cambio. Para especificar el patrón al que se aplica la plantilla se emplea el atributo *match*, cuyo valor es una expresión **XPath** que define los elementos del documento XML que se verán afectados por la plantilla.

Por ejemplo, sea el siguiente documento XML que contiene información relativa al expediente académico de un determinado estudiante.

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet href="EjemploXSLT_2.xsl" type="text/xsl"?>
<expediente alumno="José Rodriguez García">
  <asignatura id="1">
    <nombre>Programación</nombre>
    <nota>Notable</nota>
  </asignatura>
  <asignatura id="2">
    <nombre>Sistemas operativos</nombre>
    <nota>Excelente</nota>
  </asignatura>
  <asignatura id="3">
    <nombre>Diseño de interfaces</nombre>
    <nota>Suspenso</nota>
  </asignatura>
  <asignatura id="4">
    <nombre>Bases de datos</nombre>
    <nota>Bien</nota>
  </asignatura>
</expediente>
```

La siguiente hoja de estilo XSLT asociada, cuyo archivo se denomina *EjemploXSLT_2.xsl*, transforma el documento XML anterior en un documento HTML de salida.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Ejemplo</title>
        <style>
          .tabla { display: table; border: 2px solid black; }
          .fila { display: table-row; }
          .filaEncabezado { display: table-row; background-color:#9acd32;
                            font-weigth: bold; }
          .celda { display: table-cell; border: 1px solid black; padding:3px; }
```

```
        </style>
      </head>
      <body>
        <h2>Expediente académico</h2>
        <h3> Alumno: <xsl:value-of select="expediente/@alumno"></xsl:value-of> </h3>
        <div class="tabla">
          <div class="filaEncabezado">
            <div class="celda">Asignatura</div>
            <div class="celda">Nota</div>
          </div>
          <xsl:for-each select="expediente/asignatura">
            <div class="fila">
              <div class="celda">
                <xsl:value-of select="nombre"/>
              </div>
              <div class="celda">
                <xsl:value-of select="nota"/>
              </div>
            </div>
          </xsl:for-each>
        </div>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

En el código anterior, puede apreciarse que se ha definido una única plantilla. En efecto, solo se dispone de un elemento <xsl:template> que se aplica al elemento raíz del documento XML, tal como especifica el valor del atributo match="/". También puede apreciarse cómo se especifica el estilo de la composición tabular del documento HTML resultante, mediante el uso de propiedades de estilo CSS que se aplican a los elementos HTML que incorpora la plantilla definida. La funcionalidad y el uso del resto de los elementos XSLT que aparecen en el código, más concretamente de los elementos <xsl:value-of> y <xsl:for-each>, se analizará a continuación.

Al abrir el documento XML anterior, se obtiene el siguiente resultado.

Expediente académico

Alumno: José Rodriguez García

Asignatura	Nota
Programación	Notable
Sistemas operativos	Excelente
Diseño de interfaces	Suspenso
Bases de datos	Bien

Elemento <xsl:value-of>

Se utiliza para añadir a la salida, el valor del elemento o atributo que se selecciona mediante el atributo select, y de todos sus descendientes. El valor del atributo select es una expresión XPath que especifica qué elemento o atributo se extrae del contenido del documento XML.

En el código del ejemplo anterior, puede apreciarse el uso que se hace del elemento <xsl:value-of> para extraer y añadir al documento HTML de salida los valores del nombre del alumno, los nombres de las asignaturas y las notas correspondientes.

Elemento <xsl:for-each>

Permite realizar un bucle que puede utilizarse para seleccionar a cada uno de los elementos del documento XML que pertenezcan a un conjunto de nodos determinado.

En el código XSLT del ejemplo anterior, puede apreciarse que se incluye un elemento <xsl:for-each> para recorrer todas las asignaturas del expediente académico del alumno.

Elemento <xsl:sort>

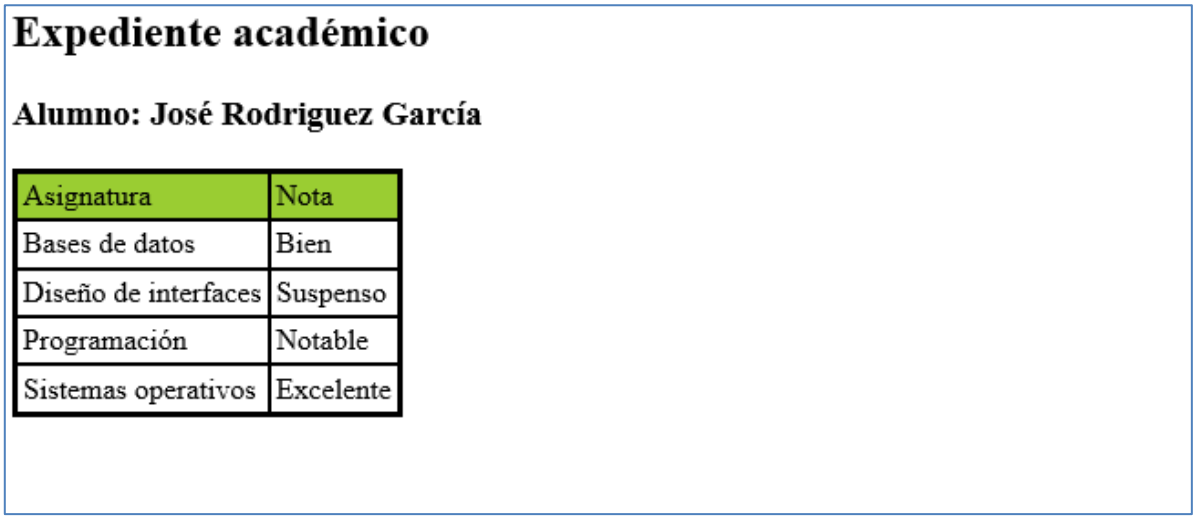
Envía a la salida, la información del documento XML de partida ordenada según un criterio de ordenación. El atributo select permite indicar el elemento por el cual se realizará la ordenación. El atributo order permite definir el orden ascendente *(ascending)* o descendente *(descending)*.

Continuando con el ejemplo anterior, puede incluirse un elemento <xsl:sort> dentro del elemento <xsl:for-each> de la plantilla de la hoja de estilos XSLT, para obtener el expediente académico del alumno ordenado, alfabéticamente y de forma descendente, por el nombre de las asignaturas.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Ejemplo</title>
        <style>
          .tabla { display: table; border: 2px solid black; }
          .fila { display: table-row; }
          .filaEncabezado { display: table-row; background-color:#9acd32;
                            font-weigth: bold; }
          .celda { display: table-cell; border: 1px solid black; padding:3px; }
        </style>
      </head>
      <body>
        <h2>Expediente académico</h2>
        <h3> Alumno: <xsl:value-of select="expediente/@alumno"></xsl:value-of> </h3>
        <div class="tabla">
          <div class="filaEncabezado">
            <div class="celda">Asignatura</div>
            <div class="celda">Nota</div>
          </div>
          <xsl:for-each select="expediente/asignatura">
            <xsl:sort select="nombre"/>
```

```
            <div class="fila">
              <div class="celda">
                <xsl:value-of select="nombre"/>
              </div>
              <div class="celda">
                <xsl:value-of select="nota"/>
              </div>
            </div>
          </xsl:for-each>
        </div>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Aplicando la hoja de estilos XSLT anterior, se obtiene el siguiente resultado.



Expediente académico

Alumno: José Rodriguez García

Asignatura	Nota
Bases de datos	Bien
Diseño de interfaces	Suspenso
Programación	Notable
Sistemas operativos	Excelente

Elemento <xsl:if>

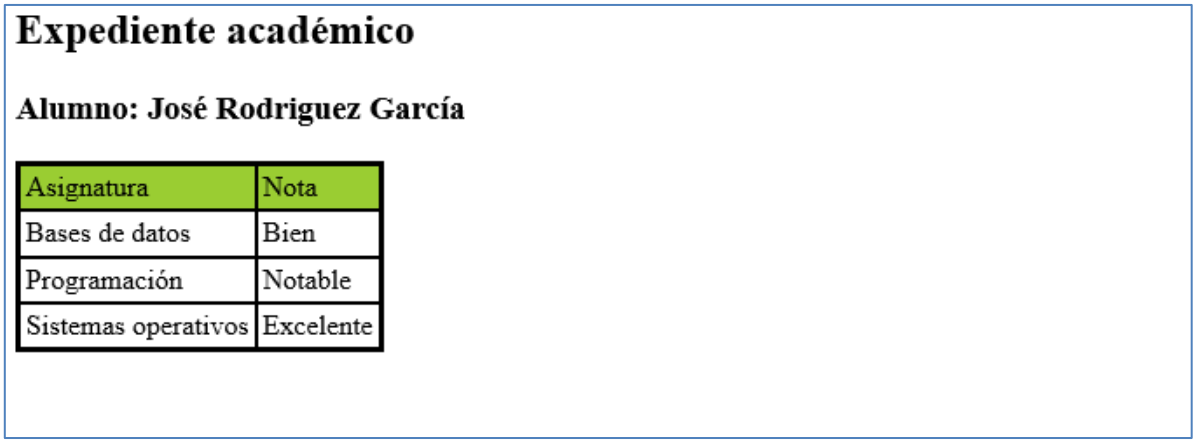
Se utiliza para producir un comportamiento condicional. Permite aplicar una plantilla sólo en el caso de que la condición especificada sea verdadera. El elemento <xsl:if> solo permite incorporar una única condición y actuar de una determinada manera cuando la condición se cumple.

Continuado con el ejemplo anterior, la siguiente hoja de estilos XSL incluye un elemento <xsl:if> dentro del elemento <xsl:for-each> para obtener solo las asignaturas aprobadas.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Ejemplo</title>
        <style>
          .tabla { display: table; border: 2px solid black; }
          .fila { display: table-row; }
          .filaEncabezado { display: table-row; background-color:#9acd32;
                            font-weigth: bold; }
```

```
          .celda { display: table-cell; border: 1px solid black; padding:3px; }
        </style>
      </head>
      <body>
        <h2>Expediente académico</h2>
        <h3> Alumno: <xsl:value-of select="expediente/@alumno"></xsl:value-of> </h3>
        <div class="tabla">
          <div class="filaEncabezado">
            <div class="celda">Asignatura</div>
            <div class="celda">Nota</div>
          </div>
          <xsl:for-each select="expediente/asignatura">
            <xsl:sort select="nombre"/>
            <xsl:if test="nota !='Suspenso'">
              <div class="fila">
                <div class="celda">
                  <xsl:value-of select="nombre"/>
                </div>
                <div class="celda">
                  <xsl:value-of select="nota"/>
                </div>
              </div>
            </xsl:if>
          </xsl:for-each>
        </div>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Aplicando la hoja de estilos XSLT anterior, se obtiene el siguiente resultado.



Expediente académico

Alumno: José Rodriguez García

Asignatura	Nota
Bases de datos	Bien
Programación	Notable
Sistemas operativos	Excelente

Elemento <xsl:choose>

El elemento <xsl:choose>, junto con los elementos <xsl:when> y <xsl:otherwise>, permite expresar un comportamiento condicional múltiple que incluya varias opciones y una opción predeterminada.

Por ejemplo, sea el siguiente documento XML que contiene información relativa al expediente académico de un estudiante. En este nuevo ejemplo, puede apreciarse que las notas son numéricas.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="EjemploXSLT_3.xsl"?>
<alumno nombre="Antonio Sala Monfort">
   <numero_expediente>A1267</numero_expediente>
   <expediente>
      <asignatura>
             <nombre>Matemáticas</nombre>
             <nota>4.00</nota>
      </asignatura>
      <asignatura>
             <nombre>Lengua</nombre>
             <nota>7.00</nota>
      </asignatura>
      <asignatura>
             <nombre>Historia</nombre>
             <nota>9.50</nota>
      </asignatura>
      <asignatura>
             <nombre>Física y Química</nombre>
             <nota>5.00</nota>
      </asignatura>
   </expediente>
</alumno>
```

La siguiente hoja de transformación XSL asociada, cuyo archivo se denomina *EjemploXSLT_3.xsl*, transforma el documento XML anterior en un documento HTML. Puede apreciarse que se utilizan las etiquetas <xsl:choose>, <xsl:when> y <xsl:otherwise> para definir una estructura condicional múltiple que permite incorporar información textual para cada una de las notas. Además, puede observarse que se emplean las secuencias de escape > y < para representar los caracteres de mayor que, '>', y de menor que, '<', en las condiciones que se incluyen en el atributo test.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
    <head>
      <style>
        .tabla { display:table; border:2px solid black; font-family:Arial; width:30% }
        .fila { display:table-row; }
        .filaEncabezado { display:table-row; background-color:#9acd32;
                          font-weight:bold; }
        .celda { display:table-cell; border:1px solid black; padding:3px; }
      </style>
    </head>
    <body>
           <h2>Expediente académico</h2>
           <h3>Alumno: <xsl:value-of select="alumno/@nombre"/></h3>
           <div class="tabla">
              <div class="filaEncabezado">
                 <div class="celda">Asignatura</div>
                 <div class="celda">Nota</div>
            <div class="celda">Descripción</div>
              </div>
              <xsl:for-each select="alumno/expediente/asignatura">
            <div class="fila">
               <div class="celda">
                  <xsl:value-of select="nombre"/>
```

```
                </div>
                <div class="celda">
                   <xsl:value-of select="nota"/>
                </div>
                <div class="celda">
                  <xsl:choose>
                     <xsl:when test="./nota &lt; 5">
                       <span style="color:red;">Suspenso</span>
                     </xsl:when>
                     <xsl:when test="./nota &gt;= 5 and ./nota &lt; 7">
                        Aprobado
                     </xsl:when>
                     <xsl:when test="./nota &gt;= 7 and ./nota &lt; 9">
                        Notable
                     </xsl:when>
                     <xsl:otherwise>
                        Excelente
                     </xsl:otherwise>
                  </xsl:choose>
                </div>
             </div>
               </xsl:for-each>
            </div>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

El ejemplo anterior, produce el siguiente resultado en el navegador.

Expediente académico

Alumno: Antonio Sala Monfort

Asignatura	Nota	Descripción
Matemáticas	4.00	Suspenso
Lengua	7.00	Notable
Historia	9.50	Excelente
Física y Química	5.00	Aprobado

Elemento <xsl:apply-templates>

El elemento <xsl:apply-templates> es, junto con el elemento <xsl:template>, uno de los elementos importantes del lenguaje de transformaciones XSLT. Indica al procesador XSLT que encuentre la plantilla a aplicar, según el tipo y el contexto de cada nodo seleccionado. La plantilla **se aplica al elemento actual o a los elementos hijos del elemento actual**. El atributo select permite especificar los nodos que se seleccionan para ser procesados, mediante una expresión XPath. Si el atributo select se omite, entonces todos los nodos descendientes del nodo actual serán procesados.

Por ejemplo, sea el siguiente documento XML que contiene información relativa al expediente académico de una alumna.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="EjemploXSLT_4.xsl"?>
<alumno nombre="Marina Sanchéz Borrell">
  <numero_expediente>A1267</numero_expediente>
  <curso>2016/2017</curso>
  <expediente>
    <asignatura>
      <nombre>Matemáticas</nombre>
      <nota>3.00</nota>
    </asignatura>
    <asignatura>
      <nombre>Lengua</nombre>
      <nota>6.00</nota>
    </asignatura>
    <asignatura>
      <nombre>Historia</nombre>
      <nota>8.50</nota>
    </asignatura>
    <asignatura>
      <nombre>Física y Química</nombre>
      <nota>5.50</nota>
    </asignatura>
  </expediente>
</alumno>
```

La siguiente hoja de transformación XSL asociada, cuyo archivo se denomina *EjemploXSLT_4.xsl*, transforma el documento XML anterior en un documento HTML.

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

    <xsl:template match="/">
      <html>
        <title>Ejemplo</title>
        <body>
          <h2>Expediente académico</h2>
          <h3>Alumno: <xsl:value-of select="alumno/@nombre"/></h3>
          <xsl:apply-templates/>
        </body>
      </html>
    </xsl:template>

    <xsl:template match="curso">
      Curso académico: <xsl:value-of select="."/><hr />
    </xsl:template>

    <xsl:template match="numero_expediente">
      Expediente: <xsl:value-of select="."/><br />
    </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">
      Asignatura: <xsl:value-of select="."/>
      <br />
    </xsl:template>

    <xsl:template match="nota">
      Nota: <span style="font-weight:bold;">
        <xsl:value-of select="."/>
      </span>
      <br />
    </xsl:template>

</xsl:stylesheet>
```

En el ejemplo anterior, puede apreciarse que en la etiqueta <xsl:apply-templates/> que aparece en la plantilla correspondiente al elemento raíz, no se especifica el atributo select, lo que provoca que se procesen todos los elementos descendientes del elemento raíz del documento XML. Además, puede observarse que se utilizan elementos <xsl:template> para especificar la plantilla de transformación correspondiente a cada uno de los elementos que deben ser procesados. También, puede observarse que la vinculación entre el elemento <xsl:apply-templates/> y el elemento <xsl:template> correspondiente a cada elemento, se establece a través del nombre del elemento que se especifica en el valor del atributo select y del atributo match, respectivamente.

El ejemplo anterior, produce el siguiente resultado en el navegador.

Como puede apreciarse en el ejemplo anterior, las características específicas de uso del elemento <xsl:apply-templates> hacen que no sea necesario incluir bucles de repetición del tipo <xsl:for-each> para recorrer el conjunto de nodos a procesar en cada caso. De manera que, el uso del elemento <xsl:apply-templates> da lugar a una organización de la hoja de estilos XSLT mucho más modular, simple y legible que facilita su creación y mantenimiento. Por estos motivos se suele recomendar el uso del elemento <xsl:apply-templates> para crear hojas de estilo XSLT.

Reglas de resolución de conflictos en la aplicación de plantillas

Debe tenerse siempre presente que el orden de aplicación del procesado XSLT queda marcado por el recorrido del árbol de nodos del documento XML de origen. El procesador XSLT sólo aplica una plantilla o regla a cada nodo. Si se han definido dos o más plantillas o reglas para un mismo nodo, el procesador XSLT solo aplica la última plantilla en aparecer. Salvo en el caso en el que se hayan definido varias reglas para un mismo nodo, el orden en el que aparecen las plantillas en la hoja de trasformaciones XSLT no es representativo.

Se han estudiado los elementos XSLT básicos y más utilizados, dado que no es posible abordar un estudio más completo. Los contenidos de este tema pueden servir para comprender algunos de los aspectos más importantes de la transformación de documentos XML. Y también, servirán para poder profundizar en su estudio, en el caso de que fuera necesario. Puede obtenerse una referencia más amplia sobre los elementos XSLT y sus funcionalidades visitando las siguientes direcciones Web:

- W3schools: https://www.w3schools.com/xml/xsl_intro.asp
- MSDN de Microsoft: https://msdn.microsoft.com/es-es/library/ms256058(v=vs.120).aspx

12.5 EL PROCESADOR XSLT. HERRAMIENTAS ESPECÍFICAS

Un procesador XSLT es una aplicación informática que procesa un documento XML mediante una hoja de transformaciones XSLT. Cuando el procesador XSLT lee un documento XML genera una representación de dicho documento como un árbol de nodos. El procesador XSLT va recorriendo y procesando nodo a nodo sobre el documento XML de origen. El nodo que se está tratando en un momento dado se denomina nodo de contexto o nodo actual.

Los procesadores XSLT se pueden clasificar en dos tipos: en línea e instalables. Algunos de los procesadores XSLT en línea más utilizados pueden ser los siguientes:

- http://online-toolz.com/tools/xslt-transformation.php
- http://www.shell-tools.net/index.php?op=xslt

Como procesadores XSLT instalables se puede mencionar Xalan (http://xml.apache.org/xalan-j/). Y algunos editores de código que incluyen un procesador XSLT que permiten realizar transformaciones XSLT como, por ejemplo: Altova StyleVision, <oXygen/>, XML Copy Editor, etc.

Por otra parte, los navegadores Web actuales incorporan un procesador XSLT, de manera que cuando se abre un documento XML que tiene asociado una hoja de estilos XSLT, se aplican las transformaciones XSLT definidas y se genera el documento de salida HTML correspondiente sobre la ventana de visualización del navegador. De modo que basta con utilizar un navegador moderno, por lo que no es necesario utilizar otras herramientas específicas.

Proceso de transformación XSLT

En este tema se estudia, básicamente, la transformación XSLT desde un documento XML a un documento HTML de salida. Este tipo de trasformaciones son las más comunes, porque facilitan la presentación visual de la información contenida en un documento XML. Para realizar este tipo de transformaciones se precisan dos documentos: el documento XML de partida y la hoja de estilos XSLT. Una vez que se dispone de estos dos documentos, una transformación XSL se puede llevar a cabo en tres ubicaciones distintas:

- En el servidor Web. Empleando para ello un procesamiento basado en lenguaje de servidor, como: PHP, ASP.NET, JSP, etc. El procesamiento en el servidor Web permite aplicar la hoja de estilos XSLT al documento XML y enviar el documento HTML al cliente como respuesta.

- En el cliente Web. El cliente Web es un navegador Web, como: Mozilla Firefox, Microsoft Edge, Google Chrome, etc. Los navegadores Web incorporan un procesador XSLT que realiza las transformaciones al abrir un documento XML que tiene asociada una hoja de estilo XSLT.

- En una aplicación específica. Pueden emplearse diversos lenguajes de programación para crear una aplicación específica que realice una transformación XSLT requerida. Igualmente, también pueden utilizarse programas genéricos para realizar las transformaciones XSLT.

El proceso de transformación XSLT consta de varias etapas que son las siguientes:

1. La hoja de transformaciones XSLT es analizada.

2. El documento XML es procesado y convertido en una estructura de árbol.

3. El procesador XSLT se posiciona en la raíz del documento XML que es el contexto inicial.

4. Los elementos que no formen parte del espacio de nombres de prefijo xsl:, como pueden ser las etiquetas HTML, se pasan directamente al flujo de salida, sin realizar sobre ellos ningún tipo de conversión. Estos elementos se denominan elementos de resultado literal.

5. Se aplica el procesado XSLT según el recorrido del árbol de nodos del documento XML. El procesador XSLT sólo aplica una plantilla o regla a cada nodo. Si se han definido dos o más plantillas o reglas para un mismo nodo, el procesador XSLT sólo aplica la última plantilla en aparecer. Salvo en este caso, el orden en el que aparecen especificadas las plantillas en la hoja de trasformaciones XSLT es irrelevante.

Glosario de términos

Árbol de nodos. Un árbol es una estructura de datos jerárquica. A su vez, un árbol es un caso particular de grafo. Los grafos permiten estudiar las interrelaciones existentes entre los elementos que interactúan entre sí unas con otras dentro de un conjunto. Estos conceptos están definidos en la teoría de grafos. Los grafos son ampliamente utilizados en la disciplina informática y su aplicación es particularmente útil cuando se emplean junto con los lenguajes de consulta, como XPath, para acceder a la información estructurada de forma jerárquica contenida en los documentos XML.

Estructura jerárquica de los documentos XML. Los tipos de nodo que pueden aparecer en un documento XML son: elemento, atributo, texto, comentario, instrucción de procesamiento y espacio de nombres. Las relaciones entre los nodos de un documento XML son similares a las que pueden encontrarse en un árbol genealógico: padres, hijos, ascendientes, descendientes, hermanos, etc. Los comentarios y las instrucciones de procesamiento son parte del árbol de nodos y deben ser tenidos en cuenta a la hora de contar nodos o iterar sobre ellos.