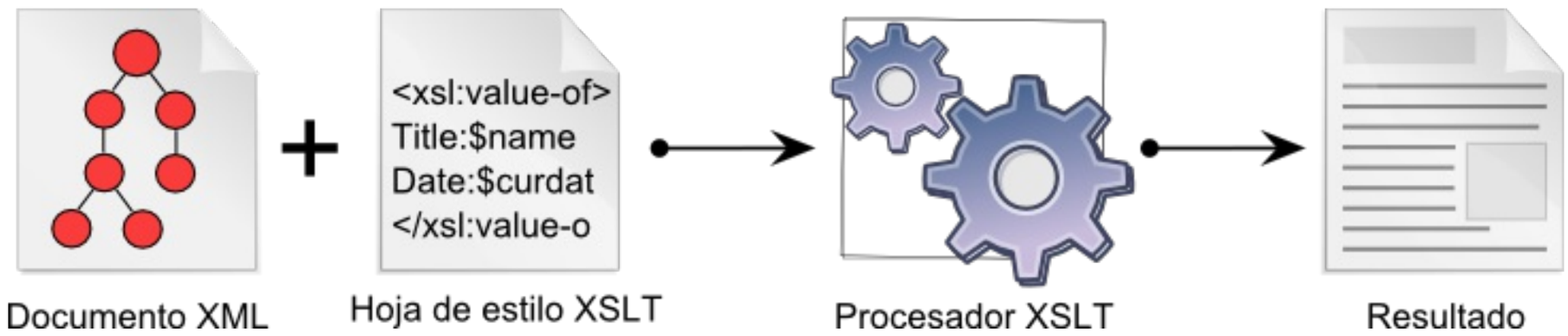
XSLT: Transformaciones XSL

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 lección 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:

- 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.

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 sutituyen 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.

Abrir documentos XML con hojas de estilo XSLT en el navegador

Al abrir en un navegador una página XML enlazada con una hoja de estilo XSLT, el navegador muestra el resultado de la transformación. Pero no muestra el código fuente obtenido como resultado, sino interpretando ese código fuente, como cuando se enlaza una hoja de estilo CSS vacía.

En los ejemplos de esta página se ha incluido a la derecha un enlace para abrir el ejemplo en el navegador. En los primeros ejemplos, el resultado es texto que se muestra todo seguido. Más adelante, en esta misma lección, en los ejemplo se generan resultados que el navegador puede mostrar en forma de párrafos. En los ejercicios se proponen transformaciones que el navegador puede mostrar en forma de listas o tablas.

Realmente, tanto en los ejemplos como en los ejercicios no es necesario abrir los archivos en el navegador. Basta con comprobar que se obtiene el resultado deseado en XML Copy Editor (u otro editor que estemos utilizando).

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>
```

Si los ejemplos de esta página se abren en el navegador, el resultado no coincide en casi todos los casos con el que se muestra en esta página ya que los navegadores no respetan los saltos de línea ni los espacios en blanco, ni muestran las etiquetas. Los resultados que se muestran en esta página se pueden obtener con XML Copy Editor.

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.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

    La vida
está en otra
parte
    Milan
Kundera


Pantaleón y
las
visitadoras
    Mario
Vargas Llosa


Conversación
en la
catedral
    Mario
Vargas Llosa
```

- 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.

  XSLT

  Resultado

  Enlace

```
<?xml
```

```
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="autor">


</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

    La vida
está en otra
parte


Pantaleón y
las
visitadoras


Conversación
en la
catedral
```

- 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.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 match="/">

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>
```

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>.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="libro">


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


</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

  Milan
Kundera
  Mario
Vargas Llosa
  Mario
Vargas Llosa
```

- En el ejemplo siguiente, se obtienen el titulo y el autor de los libros, pero uno a continuación de otro. Los saltos de línea se crean tras cada aplicación de la regla (es decir, a cada libro), pero no en el interior de la regla.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="libro">


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

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


</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

  La vida
está en otra
parteMilan
Kundera
  Pantaleón
y las
visitadorasMario
 Vargas
Llosa

Conversación
en la
catedralMario
 Vargas
Llosa
```

- En el ejemplo siguiente, los autores se obtienen gracias a la regla que extrae el contenido del nodo (el carácter 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.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="autor">


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

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

    La vida
está en otra
parte
    Milan
Kundera


Pantaleón y
las
```

```
visitadoras
    Mario
Vargas Llosa


Conversación
en la
catedral
    Mario
Vargas Llosa
```

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.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="fechaPublicacion">


<xsl:value-
of
select="@año"/>


</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

    La vida
está en otra
parte
    Milan
Kundera
    1973


Pantaleón y
las
visitadoras
    Mario
Vargas Llosa
    1973


Conversación
en la
catedral
    Mario
Vargas Llosa
    1969
```

Generar texto adicional

Se puede generar texto escribiendolo 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>.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="libro">

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

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

<p>Milan
Kundera</p>
<p>Mario
Vargas
Llosa</p>
<p>Mario
Vargas
Llosa</p>
```

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 sigue sin verse bien en el navegador, porque aunque hay etiquetas <p>, falta la etiqueta global <html>.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="libro">

     <p>
```

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

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

<p>Milan
Kundera</p>
<p>La vida
está en otra
parte</p>

<p>Mario
Vargas
Llosa</p>
<p>Pantaleón
y las
visitadoras</p>


<p>Mario
Vargas
Llosa</p>
<p>Conversación
 en la
catedral</p>
```

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

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="libro">

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



<xsl:template
 m
atch="libro">

     <p>
<xsl:value-
of
```

```
select="titulo"/>
</p>

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

<p>La vida
está en otra
parte</p>

<p>Pantaleón
y las
visitadoras</p>


<p>Conversación
 en la
catedral</p>
```

XSLT

Resultado

Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="libro">

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

</xsl:template>



<xsl:template
 m
atch="libro">

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


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

<p>Milan
Kundera</p>
```

```
<p>Mario
Vargas
Llosa</p>

<p>Mario
Vargas
Llosa</p>
```

¿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.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="libro">

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

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>

<p>Milan
Kundera
escribió "La
vida está en
otra parte"
</p>

<p>Mario
Vargas Llosa
escribió
"Pantaleón y
las
visitadoras"
</p>

<p>Mario
Vargas Llosa
escribió
"Conversación
 en la
catedral"
</p>
```

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 sí que se puede ver en el navegador ya que se interpreta como html.

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



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

<h1>Autores</h1>


<xsl:apply-
templates />
    </html>

</xsl:template>



<xsl:template
 m
atch="libro">


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

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<html>
<h1>Autores</h1>

  <p>Milan
Kundera</p>
  <p>Mario
Vargas
Llosa</p>
  <p>Mario
Vargas
Llosa</p>
</html>
```

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.

Saltos de línea y espacios en blanco: las instrucciones `<xsl:text>` y `<xsl:strip-space>`

Al transformar un documento, los procesadores XSLT incorporan saltos de línea y espacios en blanco en el resultado, pero no lo hacen

de forma uniforme. Por ejemplo, XML Copy Editor y Notepad++ (con el plug-in XML Tols) producen diferentes resultados.

No parece haber una solución sencilla que funcione en todos los procesadores, pero sí soluciones que funcionen en cada uno de ellos.

La instrucción `<xsl:strip-space>`

En el caso de XML Copy Editor, la forma más sencilla de mejorar el formato de presentación de los resultados, eliminando líneas en blanco innecesarias y sangrando los elementos anidados, es utilizar la instrucción `<xsl:strip-space>`. Pero debe tenerse en cuenta que esta instrucción no produce el mismo resultado en otros procesadores XSLT (como en Notepad++ con XML Tools).

La instrucción `<xsl:strip-space>` permite indicar si los elementos que contienen únicamente espacios en blanco se incluyen en la transformación o no.

- En el ejemplo anterior (del apartado sobre la instrucción `<xsl:apply-templates>`) la etiqueta <h1> se generaba en la misma línea que la etiqueta <html>, pero en el ejemplo siguiente se generan en líneas distintas (y las etiquetas se muestran sangradas) al utilizar la instrucción `<xsl:strip-space>`:

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">


<xsl:strip-
space
elements="*"
/>


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

<h1>Autores</h1>


<xsl:apply-
templates />
    </html>

</xsl:template>



<xsl:template
 m
atch="libro">

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

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<html>

<h1>Autores</h1>

  <p>Milan
Kundera</p>
  <p>Mario
Vargas
Llosa</p>
  <p>Mario
```

```
Vargas
Llosa</p>
</html>
```

La instrucción `<xsl:text>`

En el caso de Notepad++ con XML Tools se puede mejorar el formato de presentación de los resultados, insertando líneas en blanco innecesarias y sangrando los elementos anidados, utilzando la instrucción `<xsl:text>`. Pero debe ternerse en cuenta que esta instrucción no permite eliminar líneas en blanco que se producen en otros procesadores (como en XML Copy Editor)./p>

La instrucción `<xsl:text>` permite generar texto que no se puede generar simplemente añadiéndolo (saltos de líneas y espacios en blanco, por ejemplo).

- En el ejemplo anterior (del apartado sobre la instrucción `<xsl:apply-templates>`) la etiqueta <h1> se generaba en la misma línea que la etiqueta <html>, pero en el ejemplo siguiente se generan en líneas distintas al añadir un salto de línea con la entidad de carácter
 (y un par de espacios para aliner la etiqueta <h1> con las etiquetas <p>):

XSLT

Resultado

Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



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

<xsl:text>&#10;

</xsl:text>

<h1>Autores</h1>


<xsl:apply-
templates />
    </html>

</xsl:template>



<xsl:template
 m
atch="libro">

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

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<html>

<h1>Autores</h1>

  <p>Milan
Kundera</p>
  <p>Mario
```

```
Vargas
Llosa</p>
  <p>Mario
Vargas
Llosa</p>
</html>
```

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:

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="licencia">

    <p><img
src="
<xsl:value-
of
select="imagen"
 />" /></p>

</xsl:template>


</xsl:stylesheet>
```

```
Error at
line 5,
column 19:
not well-
formed
(invalid
token)
```

  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:

XSLT

Resultado

Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



<xsl:template
 m
atch="licencia">

    <p><img>

<xsl:attribute
 name="src">

<xsl:value-
of
select="imagen"
 />

</xsl:attribute>

    </img>
    </p>

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<p>
  <img
src="cc_bysa_88x31.png"/>

</p>
```

En la hoja de estilo XSLT, la etiqueta <img> se escribe con apertura y cierre, aunque en el resultado aparece como etiqueta monoatómica.

De todas formas, el navegador no mostraría todavía la imagen, puesto que no interpreta la etiqueta <img> como la etiqueta de imagen del html.

- Como en ejemplos anteriores, para que la imagen se muestre en el navegador sería necesario generar también la etiqueta <html>:

  XSLT

  Resultado

  Enlace

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 v
ersion="1.0">



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

<xsl:apply-
```

```
templates />
    </html>

</xsl:template>



<xsl:template
 m
atch="licencia">

    <p><img>

<xsl:attribute
 name="src">

<xsl:value-
of
select="imagen"
 />

</xsl:attribute>

    </img>
    </p>

</xsl:template>


</xsl:stylesheet>
```

```
<?xml
version="1.0"

encoding="UTF-
8"?>
<html>
  <p><img
src="cc_bysa_88x31.png"/>
</p>
</html>
```