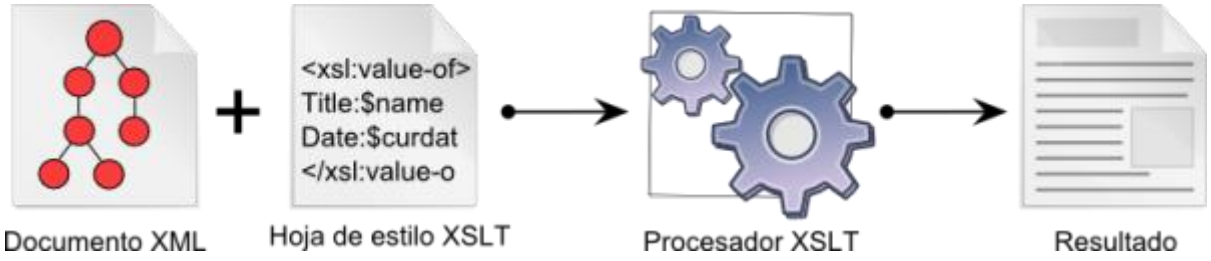
Qué es el lenguaje de transformación XSLT

Introducción a XSLT

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

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

El **funcionamiento** lo podemos observar en la siguiente imagen:



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

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

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

XSLT es un estándar del **W3C**:

- XSLT 1.0
- XSLT 2.0
- XSLT 3.0

Para todos los **ejemplos** siguiente vamos a tomar como referencia el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>

<bib url="http://www.ticarte.com">

  <book id="1">

    <title>TCP/IP Illustrated</title>

    <author>Stevens</author>

    <publisher>Addison-Wesley</publisher>

    <year>2002</year>

  </book>

  <book id="2">

    <title>Advanced Programming in the Unix Environment</title>

    <author>Stevens</author>

    <publisher>Addison-Wesley</publisher>

    <year>2004</year>

  </book>

  <book id="3">

    <title>Data on the Web</title>

    <author>Abiteboul</author>

    <author>Buneman</author>
```

```
    <author>Suciu</author>

    <year>2006</year>

  </book>

</bib>
```

Software y herramientas online

Para diseñar hojas de estilos XSLT podemos utilizar el **software** libre XML Copy Editor o cualquier otro software que comentamos aquí.

Para probarlo sólo tenemos que abrir el documento XML en el navegador **Mozilla Firefox o Internet Explorer** para ver el resultado de la transformación. El navegador Google Chrome por defecto no aplica las transformaciones.

En el navegador podemos utilizar la opción del botón derecho "**Inspeccionar elemento**" para comprobar los resultados de la transformación, ya que si elegimos "Ver código fuente" se nos abrirá el documento XML original sin ninguna transformación.

Aplicar al documento XML una hoja de estilos XSLT

El primer paso será definir en la **cabecera del documento XML** qué hoja de estilos XSLT se va a aplicar como vemos a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>

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

```
<bib>
```

Este documento XML podemos abrirlo en los navegadores que hemos indicado anteriormente.

Recordamos que las hojas de estilos XSLT deben estar **bien formadas**, pero no van a validar.

Estructura básica de una hoja de estilos XSLT

La estructura básica de un documento XSLT es la que se muestra a continuación. Al no contener ninguna regla de transformación todos los nodos se enviarán por defecto a la salida de la transformación.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

</xsl:stylesheet>
```

<xsl:output>

En su interior podemos utilizar la etiqueta <xsl:output> que nos indicará el tipo de documento al que vamos a transformar el documento XML de entrada. Se permiten los siguientes atributos:

- method: define el formato de salida (xml, html o text).
- version: define la versión del formato de salida.
- enconding: juego de caracteres de salida. Por defecto UTF-8.
- indent: indenta la salida de la transformación (yes o no).

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <xsl:output method="html" version="4.0" encoding="UTF-8"
indent="yes" />

</xsl:stylesheet>
```

Plantillas

<xsl:template> / <xsl:apply-templates> / <xsl:value-of>

La transformación se basa en el uso de plantillas (templates) con la etiqueta <xsl:template> que se aplicarán a cada nodo o atributo del documento según se va recorriendo éste, sustituyendo de esa manera el nodo por el contenido de su plantilla.

La hoja de estilos debe tener al menos una plantilla que suele corresponder con el nodo raíz, aunque puede ser de otro nodo. Si esta plantilla no contiene nada, como en el siguiente ejemplo, no se enviará nada como salida de la transformación:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <xsl:output method="html" version="4.0" encoding="UTF-8"
indent="yes" />

  <xsl:template match="/">

  </xsl:template>

</xsl:stylesheet>
```

***NOTA**: En los siguientes ejemplos eliminaremos las etiquetas <xsl:stylesheet> y <xsl:output> para abreviar los ejemplos.*

Lo normal es que tengamos más plantillas, una para cada nodo de nuestro documento XML. Utilizaremos la etiqueta <xsl:apply-templates> para mostrar en una plantilla la transformación de otra. Prestar atención a los atributos que se utilizan, en <xsl:template> se utiliza "match" y en <xsl:apply-templates> se utiliza "select".

En el siguiente ejemplo la plantilla raíz transformará cada nodo "book" de su interior por su plantilla. Hay que ver como las plantillas <xsl:template> utilizan rutas XPath absolutas para identificar los nodos a los que hacen referencia, pero <xsl:apply-templates> utilizará, normalmente, rutas XPath relativas al nodo de contexto en el que se esté llamando. Este ejemplo aún no muestra nada en la salida.

```
<!-- Plantilla raíz -->

<xsl:template match="/">

  <xsl:apply-templates select="bib/book" />

</xsl:template>



<!-- Plantilla book -->

<xsl:template match="/bib/book">

</xsl:template>
```

En el interior de una plantilla podemos utilizar la etiqueta <xsl:value-of> para imprimir el valor que contiene el nodo, utilizando expresiones XPath para acceder a ellas.

Tomaremos como **nodo contexto** el nodo en el que estamos trabajando o estamos posicionados actualmente. El ejemplo anterior lo podemos modificar mostrando en la plantilla "book" el contenido del nodo "title", para mostrar todos los títulos de los libros:

```
<!-- Plantilla raíz -->

<xsl:template match="/">

  <xsl:apply-templates select="bib/book" />

</xsl:template>



<!-- Plantilla book -->

<xsl:template match="/bib/book">

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

</xsl:template>
```

En el ejemplo anterior hemos mostrado el valor del nodo "title" desde "book", pero podemos crear también una nueva plantilla para el nodo "title" y que dicha plantilla muestre directamente su valor de la siguiente manera. Mirar de nuevo cómo se utilizan las rutas absolutas y relativas en XPath.

```
<!-- Plantilla raíz -->

<xsl:template match="/">

  <xsl:apply-templates select="bib/book" />

</xsl:template>
```

```
  <!-- Plantilla book -->

  <xsl:template match="/bib/book">

    <xsl:apply-templates select="title" />

  </xsl:template>



  <!-- Plantilla title -->

  <xsl:template match="/bib/book/title">

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

  </xsl:template>
```

Si una plantilla lo único que tiene es mostrar el valor de su nodo se podría eleminar, obteniéndose el mismo resultado. En el siguiente ejemplo eliminamos la plantilla del nodo "title" y aún así el resultado es el mismo que el ejemplo anterior:

```
<!-- Plantilla raíz -->

  <xsl:template match="/">

    <xsl:apply-templates select="bib/book" />

  </xsl:template>



  <!-- Plantilla book -->

  <xsl:template match="/bib/book">
```

```
  <xsl:apply-templates select="title" />
</xsl:template>
```

También se puede utilizar <xsl:apply-templates> sin seleccionar ningún nodo concreto, de esa manera se aplicarán todas las plantillas que existan para los nodos que se recorran. Si en el ejemplo anterior no utilizamos "select" dentro de la plantilla "book" se mostrará por salida todos los datos de los nodos que contenga dentro "book", existan o no sus plantillas.

```
<!-- Plantilla raíz -->
<xsl:template match="/">
  <xsl:apply-templates select="bib/book" />
</xsl:template>

<!-- Plantilla book -->
<xsl:template match="/bib/book">
  <xsl:apply-templates />
</xsl:template>

<!-- Plantilla title -->
<xsl:template match="/bib/book/title">
  <xsl:value-of select="." />
```

```
  </xsl:template>
```

Los atributos también pueden transformarse creándoles una plantilla como vemos en el siguiente ejemplo:

```
<!-- Plantilla raíz -->

  <xsl:template match="/">

    <xsl:apply-templates select="bib/book" />

  </xsl:template>



  <!-- Plantilla book -->

  <xsl:template match="/bib/book">

    <xsl:apply-templates select="@id" />

  </xsl:template>



  <!-- Plantilla atributo id -->

  <xsl:template match="/bib/book/@id">

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

  </xsl:template>
```

<xsl:sort>

Se puede ordenar la salida de los nodos de una plantilla mediante la etiqueta <xsl:sort> situándola dentro de las etiquetas <xsl:apply-templates>. Los siguientes atributos se pueden utilizar con ella:

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

```
<!-- Plantilla bib -->

<xsl:template match="/bib">

  <ul>

    <xsl:apply-templates select="book">

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

    </xsl:apply-templates>

  </ul>

</xsl:template>
```

Ejemplo completo

En el siguiente ejemplo más completo utilizaremos las plantillas para transformar el documento XML a una lista con formato HTML donde cada elemento de la lista sea el nombre del libro y el año de publicación entre paréntesis, situando el código HTML en sus plantillas correspondientes:

```
<!-- Plantilla raíz -->

  <xsl:template match="/">

    <html>
```

```
    <head>
    </head>
    <body>
      <xsl:apply-templates select="bib" />
    </body>
    </html>
</xsl:template>

<!-- Plantilla bib -->
<xsl:template match="/bib">
  <ul>
    <xsl:apply-templates select="book" />
  </ul>
</xsl:template>

<!-- Plantilla book -->
<xsl:template match="/bib/book">
  <li>
    <xsl:apply-templates select="title" />
```

```
    <xsl:apply-templates select="year" />
  </li>
</xsl:template>

<!-- Plantilla title -->
<xsl:template match="/bib/book/title">
  <xsl:value-of select="." />
</xsl:template>

<!-- Plantilla year -->
<xsl:template match="/bib/book/year">
  (<xsl:value-of select="." />)
</xsl:template>
```

Sólo la plantilla raíz

<xsl:for-each>

El mismo ejemplo anterior se puede realizar utilizando solamente la plantilla raíz y haciendo uso de la etiqueta <xsl:for-each>, que permite pasar por todos los nodos seleccionados, como si se tratara de un bucle "for" en programación:

```
<!-- Plantilla raíz -->
```

```
<xsl:template match="/">
  <html>
    <head>
    </head>
    <body>
      <xsl:for-each select="bib">
        <ul>
        <xsl:for-each select="book">
          <li>
          <xsl:value-of select="title" />
          </li>
        </xsl:for-each>
        </ul>
      </xsl:for-each>
    </body>
    </html>
</xsl:template>
```

Estructuras condicionales

<xsl:if>

La etiqueta <xsl:if> permite introducir una condición en la transformación mediante una condición evaluada mediante XPath en el atributo "test". Pero tan simple que no admite ni siquiera la opción "else". Hay que tener en cuenta que para utilizar los operadores "mayor que" y "menor que" hay reemplazarlos por su carácter codificado en HTML.

- Mayor que: >
- Menor que: <

```
<!-- Plantilla year -->

<xsl:template match="/bib/book/year">

  <xsl:if test="year &gt; 2000">

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

  </xsl:if>

  <xsl:if test="year &gt; 2000">

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

  </xsl:if>

</xsl:template>
```

<xsl:choose> / <xsl:when> / <xsl:otherwise>

La estructura "choose" es similar al "switch" de los lenguajes de programación, permitiendo condiciones consecutivas:

```
<!-- Plantilla year -->

<xsl:template match="/bib/book/year">
```

```
    <xsl:choose>
      <xsl:when test="year &gt; 2000">
        <strong><xsl:value-of select="." /></strong>
      </xsl:when>
      <xsl:when test="year &gt; 2000">
        <em><xsl:value-of select="." /></em>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="." />
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
```

Generar contenido

<xsl:text>

En algunas ocasiones necesitaremos texto literal en la salida, por ejemplo espacios en blanco para separar valores. En este caso utilizaremos la etiqueta <xsl:text> con su código correspondiente:

```
  <!-- Plantilla year -->
  <xsl:template match="/bib/book/year">
```

```
    (<xsl:text>&#10;</xsl:text>
     <xsl:value-of select="." />
     <xsl:text>&#10;</xsl:text>)
  </xsl:template>
```

<xsl:element> / <xsl:attribute>

Cuando tengamos que crear nodos nuevos en el documento de salida, con su etiquetas y atributos necesitaremos utilizar las etiquetas <xsl:element> y <xsl:attribute> para poder utilizar los valores del XML origen en dichos nodos. En el siguiente ejemplo crearemos enlace mediante HTML, es decir, una etiqueta "a" con el atributo "href" utilizando el atributo "url" del nodo "bib":

```
<!-- Pantilla bib -->
<xsl:template match="/bib">
  <xsl:element name="a">
    <xsl:attribute name="href">
      <xsl:value-of select="@url" />
    </xsl:attribute>
  </xsl:element>
  <ul>
    <xsl:apply-templates select="book" />
  </ul>
```

```
</xsl:template>
```