

1. Técnicas de transformación de documentos XML.

La tecnología XML permite separar de manera efectiva los datos a almacenar, la estructura o semántica en la que se organizan y la presentación de los mismos.

Necesitaremos tecnologías nuevas para transformar y mostrar los datos en el formato que deseamos. Estas herramientas se engloban dentro de **eXtensible Stylesheet Language**, XSL.

XSL es a XML, lo que las hojas de estilo en cascada (CSS) a HTML

La XSL es una especificación desarrollada dentro del W3C para aplicar formato a los documentos XML de forma estandarizada. El primer boceto de esta especificación aparece el 18 de agosto de 1998.

XSL que es un lenguaje para escribir hojas de estilo **consta de dos partes :**

- **Un lenguaje de transformación**, mediante el cual se puede transformar un documento XML en otro XML.
- **Un lenguaje de formateo**, que no es más que un vocabulario XML para especificar objetos de formateo (FO)

- Durante su desarrollo, se decidió "sacar" fuera de la especificación XSL el lenguaje de transformación y desarrollarlo como una especificación "independiente" a la que se denominó **XSLT**.

XSL hace uso de las XSLT, aunque la XSLT esta diseñada para ser utilizada independientemente de la XSL.

- Durante el desarrollo de la especificación XSLT también se decidió "sacar" fuera de ella la parte del lenguaje de transformación que hacía referencia a la forma de acceder y moverse por los nodos de un documento XML. A esta nueva especificación se le denominó **XPath**, y ha sido desarrollada no para utilizarse de forma independiente, sino desde las XSLT.

2. Transformación de documentos con XSL

Una hoja XSL es también un documento XML, como los canales RSS o los documentos XSD (Schema XML)

¿Qué **transformaciones** podemos hacer en XML usando **XSL**?

- A otro documento XML.
- A un documento HTML.
- A un documento de texto.

Si se quiere utilizar correctamente esta tecnología de representación de documentos XML se deben seguir los siguientes pasos:

- ☐ Tener validado el documento XML.
- ☐ Crear una hoja de estilo XSL bien formada.
- ☐ Vincular el documento XML y la hoja de estilo XSL.

Para vincular en un documento XML una hoja de estilo XSL cualquiera hay que incluir la siguiente línea:

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

Si se quiere utilizar correctamente esta

01ejemplo.xml

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

```
<!DOCTYPE libreria SYSTEM "01ejemplo.dtd">
```



01ejemplo.dtd

```
<?xml-stylesheet type="text/xsl" href="01ejemplo.xsl"?>
```

```
<libreria>
```

```
</librería>
```



01ejemplo.xsl

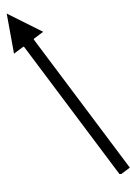
```
<?xml version="1.0" encoding="UTF-8"?>
```

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

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

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

```
</xsl:stylesheet>
```

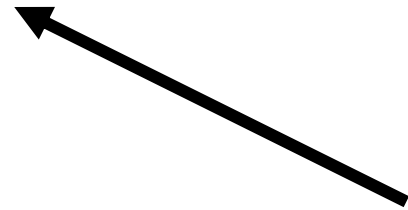


Etiqueta que identifica el fichero XSL y el espacio de nombres en el que se basa.

La etiqueta que define la plantilla a usar dentro del XSL y que engloba todas las demás etiqueta es

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

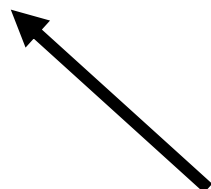
```
</xsl:template>
```



El atributo match nos permite indicar sobre qué parte del documento XML se quiere actuar. Con "/" indicamos que sobre la raíz.

Elementos básicos

<xsl:for-each **select**=“elementos_a_recorrer”>



Recorrerá todo el conjunto de elementos XML indicados

<xsl:for-each select="libreria/libro">

...

</xsl:for-each>

<xsl:for-each select="libreria/libro[autor='Carlos Garre']">

...

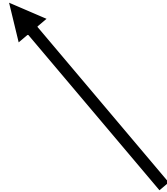
</xsl:for-each>



Recorrerá sólo los de este autor

Elementos básicos

<xsl:value-of **select**="elementos_a_extraer"/>



Extraerá la información del elemento indicado

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

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

Elementos básicos

02cdcatalogo.xml

02cdcatalogo.dtd

02cdcatalogo.xsl

<xsl:sort **select**="elementos_por_el_que_ordenar"/>



La información se ordenará por el elemento indicado.

Si no se indica nada el orden es el que aparezca en el documento XML.

```
<xsl:for-each select="catalog/cd[country='UK']">
```

```
  <xsl:sort select="artist"/>
```

```
</xsl:for-each>
```

```
  <xsl:for-each select="catalog/cd[country!='UK']">
```

```
    <xsl:sort select="artist"/>
```

```
  </xsl:for-each>
```

Operadores en XSL.

02cdcatalogo.xml

02cdcatalogo.dtd

02cdcatalogo.xsl

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 que “>” >

```
<xsl:for-each select="catalog/cd[price&lt;='9.90']">
```

Condicionales. IF

`<xsl:if test="condición"/>`



Puede emplearse también un elemento concreto para condicionar que mostrará la transformación.

`<xsl:if test="country!='UK'"/>`
`</xsl:if>`

02cdcatalogo.xml

02cdcatalogo.dtd

02cdcatalogo.xsl

02bejemplo.xml

02bejemplo.dtd

02bejemplo.xsl

Condición en atributos (@)

Condicionales. CHOOSE

Además de `<xs:if>` podemos establecer múltiples condiciones dentro del recorrido del árbol XML con `<xsl:choose>`. Similar a la instrucción *switch* de Javascript o C++.

Se pueden establecer tantas expresiones condicionales como se quieran mediante los elementos `<xsl:when>`. Puede establecerse una condición por defecto usando `<xsl:otherwise>`.

```
<xsl:choose>
```

```
    <xsl:when test="expresión1">
```

```
    </xsl:when>
```

```
    ...
```

```
    <xsl:when test="expresión2">
```

```
    </xsl:when>
```

```
</xsl:choose>
```

03ejemplo.xml

03ejemplo.dtd

03ejemplo.xsl

Agrupación por atributos

Hemos incluido en el recorrido del árbol los elementos intermedios y terminales que nos interesaban. También podemos incluir los atributos. Usando **@nombre_atributo**

```
<xsl:for-each select="productos/producto">
  <tr>
    <td><xsl:value-of select="@pid"/></td>
    <td><xsl:value-of select="name"/></td>
    <td><xsl:value-of select="price"/></td>
  </tr>
</xsl:for-each>
```

04ejemplo.xml

04ejemplo.dtd

04ejemplo.xsl

05ejemplo.xml

05ejemplo.dtd

05ejemplo.xsl

3. La transformación a HTML

Desde la aplicación XML Copy Editor podemos usando **XML -> XSL Transformation** generar el documento .html equivalente expandido y que no necesitará de un procesador XSL para ser visualizado.

Podemos comprobar como desde Internet Explorer si se transformaban directamente los .xml asociados con .xsl y usando Ver Código fuente se observa la transformación.

4. Uso de plantillas

Hasta el momento solo hemos usado una única plantilla para todo el documento XML empleando `<xsl:template>`. Podemos dividir la transformación usando diferentes plantillas según la ubicación donde se encuentre el elemento.

En lugar de recorrer el árbol indicamos como comportarse cuando encuentre determinado elemento.

Emplearemos `<xsl:apply-templates/>` para indicar se apliquen las plantillas definidas y `<`para que se aplique concretamente para ese *elemento*.

`xsl:apply-templates select="elemento"/>`

Cada plantilla se define con `<xsl:template match="elemento">`, incluyendo la raíz `<xsl:template match="/">`

06ejemplo.xml

06ejemplo.dtd

06ejemplo.xsl

08ejemplo.xml

08ejemplo.xsl

5. Variables

Mediante el elemento `<xsl:variable>` podemos definir variables.

Incluye el atributo obligatorio *name*, que toma como valor el nombre de la variable

Para asignar el valor deseado a las variables hay dos posibilidades:

1. Emplear el atributo *select*

```
<xsl:variable name="a" select="5"/>
```

```
<xsl:variable name="atrib1" select="codigo"/>
```

```
<xsl:variable name="enlace" select="centro/@web"/>
```

2. Por el contenido del elemento <xsl: variable>

```
<xsl:variable name="cabeceraTabla">
```

```
  <tr>
```

```
    <th>Element</th>
```

```
    <th>Description</th>
```

```
  </tr>
```

```
</xsl:variable>
```

```
<xsl:variable name="color">red</xsl:variable>
```

Para llamar a una variable declarada hay que usar \$ junto al nombre de la variable.

<xsl:value-of select="\$a+\$b"/> ← Declaraciones tipo1 07ejemplo.xml
07ejemplo.xsl

<xsl:value-of select="\$color"/>

<xsl:copy-of select="\$cabeceraTabla" /> ← Insertamos el elemento
referenciado en la variable,
incluidos los hijos.

Ir a enlace

← Como es el valor de un atributo debe ir entre {}