La diferencia entre los términos XSl y XSLT es en ocasiones ambigua. En teoría XSL es una familia de especificaciones que agrupa a:

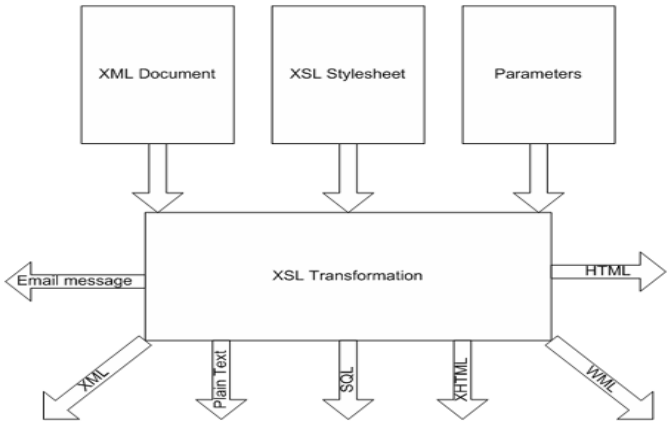
XPath (calificación de nodos dentro de un documento)
XSLT (lenguaje de transformaciones)
XSL-FO (lenguaje de presentación)

¿QUÉ ES XSLT?

- XSLT (eXtensibleStylesheetLanguage – Transformations), Lenguaje de hoja de estilo ampliable para transformaciones
- Describe un lenguaje basado en XML para transformar documentos XML a cualquier otro formato
- XSLT es el lenguaje de hojas de estilo recomendado de XML.
- XSLT es mucho más sofisticado que el CSS.
- XSLT puede ser utilizado para transformar documentos XML en HTML, antes de ser mostrados en un navegador.

Aplicación de las transformaciones

- Utilizaremos XSLT para transformar documentos entre esquemas XML que permitan su procesamiento por distintos sistemas
- Utilizaremos XSLT para transformar documentos XML en HTML, o cualquier otro formato que facilite su presentación en la pantalla de un ordenador o en impresora
- La transformación de XML a HTML es el principal uso que se hace de XSLT



- No debemos confundir las transformaciones XSLT con la presentación de documentos XML con CSS
- Con XSLT, generaremos un documento HTML a partir de un documento XML. Se tratará de *dos documentos "distintos"*

- Con CSS, el navegador recibe un documento XML que formatea utilizando las reglas CSS para presentarlo en pantalla de forma que sea más fácilmente legible, pero es el mismo documento

Estructura de una hoja de estilo XSLT

- Una hoja de estilo XSLT es un documento XML. Debe estar bien formado.
- Las hojas de estilo se guardarán siempre en archivos independientes con extensión .xsl
- Deben comenzar con una declaración XML:
  <?xml version="1.0"?>
- El elemento raíz de la hoja de estilo XSLT es stylsheet.
- Este elemento contendrá a todos los demás, y debe ir precedido por el alias xsl correspondiente al espacio de nombres para hojas de estilo XSLT.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

.....

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

- Entre las marcas de inicio y de fin del elemento raíz xsl:stylesheet, se escribirán las reglas de transformación que se llaman plantillas
- Cada regla se definirá mediante un elemento xsl:template (una plantilla)
- Se utiliza el atributo match para asociar una plantilla con un elemento XML. El atributo match también se puede utilizar para definir un modelo para todo el documento XML. El valor del atributo match es una expresión de XPath (eje. match = "/" define todo el documento).
- La regla indica qué instancias de los elementos del documento XML se van a transformar.

REGLAS DE TRANSFORMACIÓN

EJEMPLO:

```
<xsl:template match="//nombre"> ----- árbol de origen al que se aplica esta plantilla
<html>
<body>
        <h2><xsl:value-of select="." /></h2>
</body>
</html>
</xsl:template>
```

- La regla se aplicará a todas las instancias del elemento nombre. Esto se indica mediante el atributo **match** que acompaña al elemento **xsl:template**.
- Entre las etiquetas de inicio y de fin del elemento **xsl:template** se escribe la transformación que se debe realizar... es decir, qué texto y qué marcas se escribirán en

el documento resultado de la transformación, cada vez que se encuentre una instancia del elemento **nombre** en el documento origen.

- Con **<xsl:value-of...>**, se recupera y escribe en el documento resultado el valor del elemento que está siendo procesado.

Ejemplo transformación XSLT Documento XML

```
<?xmlversion="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet type="text/xsl" href="prueba.xsl" ?>----Referencia al documento XSl
<ciudades>
        <ciudad>
                <nombre>Madrid</nombre>
                <habitantes>3500000</habitantes>
        </ciudad>
        <ciudad>
                <nombre>Málaga</nombre>
                <habitantes>800000</habitantes>
        </ciudad>
        <ciudad>
                <nombre>Toledo</nombre>
                <habitantes>50000</habitantes>
        </ciudad>
</ciudades>
```

Documento XSL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="//nombre"> --------- queremos obtener los nombres de las ciudades
        <html>
        <body>
                <h2><xsl:value-of select="." /></h2>
        </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

Resultado

```
<h2>Madrid</h2>3500000
<h2>Málaga</h2>800000
<h2>Toledo</h2>50000
```

El resultado obtenido no es el documento HTML esperado.

Vemos que en el documento de salida no sólo se ha incluido el texto de los elementos procesados, sino el de todos los elementos del documento original…

Para evitar esto, tenemos que hacer unos cambios en la hoja de estilo XSLT

Documento xsl mejorado

```
<?xmlversion="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
        <html>
        <head>
                <title>Ejemplo XSLT</title>
        </head>
        <body>
                <h1> CIUDADES DE ESPAÑA </h1>
                <xsl:apply-templates select="//nombre" />
        </body>
        </html>
</xsl:template>

<xsl:template match="nombre">
        <h3><xsl:value-of select="." /></h3>
</xsl:template>
</xsl:stylesheet>
```

La regla “de inicio”

- La regla <xsl:template match=“/”> se ejecuta cuando se encuentra el elemento raíz del documento XML
- Dentro de esta regla, podemos incluir llamadas a otras reglas definidas en la hoja de estilo, mediante el elemento:

  <xsl:apply-templates select=”…” />

- El atributo select tomará como valor el nombre del elemento asociado a la regla que queremos “disparar”
- Esto nos ofrece un control real sobre el “orden” de ejecución de las reglas

El resultado de la transformación es el siguiente:

```
<html>
        <head>
                <title>Ejemplo XSLT</title>
        </head>
        <body>
                <h1> CIUDADES DE ESPAÑA </h1 >
                        <h3>Madrid</h3>
                        <h3>Málaga</h3>
                        <h3>Toledo</h3>
        </body>
</html>
```

El element <xsl:value-of...>

- En el elemento <xsl:value-of…> se puede indicar que se quiere mostrar el valor del elemento que estamos procesando
- También podemos indicar que queremos mostrar el valor de un elemento hijo, o descendiente, del elemento que se está procesando
- Esto es posible porque en el atributo select podemos utilizar una "expresión XPATH"
- XPATH es un lenguaje que nos permite direccionarnos a las secciones de un documento XML y obtener las partes de la información que deseamos (nodo de contexto)
- Por ejemplo, para mostrar el valor del elemento artículo, que es un hijo del elemento producto, podríamos utilizar la siguiente regla:

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

El valor del atributo select se puede leer de la siguiente forma: "dame el valor del elemento artículo que es hijo del elemento que estoy procesando". En este caso, cada uno de los elementos producto. Esto se indica mediante ./

Expresión	Descripción
nodename	Selecciona todos los nodos secundarios del nodo denominado
/	Selecciona desde el nodo raíz
//	Selecciona nodos en el documento desde el nodo actual que coincidan con la selección sin importar donde se encuentren
.	Selecciona el nodo actual
..	Selecciona el padre del nodo actual
@	Selecciona los atributos

Procesamiento de nodos por lotes <xsl:for-each>

El elemento <xsl:for-each> permite hacer un bucle en XSLT.
Se puede utilizar para seleccionar todos los elementos XML del nodo actual.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
    <body>
        <h2>CIUDADES DE ESPAÑA</h2>
        <table border="1">
            <trbgcolor="#9acd32">
                <th>Ciudades</th>
                <th>Habitantes</th>
```

```
                    </tr>
                    <xsl:for-each select="ciudades/ciudad">
                    <tr>
                        <td><xsl:value-of select="nombre"/></td>
                        <td><xsl:value-of select="habitantes"/></td>
                    </tr>
                    </xsl:for-each>
                </table>
        </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Orden de procesamiento

- Las reglas se van activando y ejecutando a medida que se recorre el documento origen que se quiere transformar.
- De esta forma, las reglas se ejecutan en el orden en el que se van encontrando los elementos en el documento.
- Este comportamiento por defecto puede cambiarse en las hojas de estilo XSLT, a diferencia de lo que sucedía en las hojas de estilo CSS
- Esto permite "reordenar" los contenidos del documento XML, de una forma distinta a como están ordenadas en el documento XML inicial
- Para ordenar los contenidos, se utiliza el elemento xsl:sort
- xsl:sort nos da la posibilidad de ordenar los resultados obtenidos al procesar el documento original a través de los elementos <xsl:for-each> y <xsl:apply-templates>
- Acepta varios atributos:
  select – que toma como valor el nombre del elemento que se va a utilizar como criterio de ordenación.
  order – que indica si se debe utilizar un orden ascendente o descendente.
  data-type – indica tipo de dato (number para valores numéricos), por defecto es tipo text.

```
<xsl:apply-templates select="//ciudad">
        <xsl:sort select="ciudad" order="descending" />
</xsl:apply-templates>
```

Asociar una hoja de estilo XSL a un documento

Debemos incluir, tras la declaración XML, la siguiente instrucción de procesamiento:

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

Ejemplo

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="http:/www.anaya.es/docs/xml/ejemplo.xsl"?>
        <documento>
                <titulo>Programar ASP</titulo>
                <paginas>456</paginas>
```

```
            <anno-pub>2001</anno-pub>
        </documento>
</stylesheet>
```

Leer y obtener el valor de atributos en XSLT

- En XSLT podemos "filtrar" o indicar qué instancias de un elemento queremos procesar, tomando como criterio de selección el valor de los atributos que acompañan a los elementos
- Para hacer esto, en un elemento xsl:value-of, podemos recuperar el valor de un atributo mediante la expresión @nombreAtributo, por ejemplo:

```
<libro genero="novela">
        <autor>DeepakChopra</autor>
        <titulo>El sendero del Mago</titulo>
        <isbn>950-15-1727</isbn>
        <editorial>Harmany Book</editorial>
        <sumario>… nos muestra cómo debemos … Por medio de historias como </sumario>
        <precio moneda="euros">50.00</precio>
</libro>
```

```
<xsl:template match="libro">
<tr>
<td><xsl:value-of select="@genero" /></td>
<td><xsl:value-of select="./precio/@moneda" /></td>
/tr>
</xsl:template>
```

Elemento xsl:if

El elemento <xsl:if> contiene una plantilla que se aplicará sólo si la condición especificada es verdadera.

SINTAXIS

```
<xsl:if test="expresion">
        <!–Contenido de template -->
</xsl:if>
```

EJEMPLO

```
<xsl:for-each select="//ciudad">
        <xsl:if test="habitants &gt; 100000">
                <h3><xsl:value-of select="./nombre" /></h3>
                <xsl:value-of select="./habitantes" />
        </xsl:if>
</xsl:for-each>
```

OPERADORES XSL	
Igualdad (=)	=
Desigualdad (≠)	!=
Menor que (<)	<
Mayor que (>)	>

Elemento xsl:choose

El elemento <xsl:choose> se utiliza en conjunción con <xsl:when> y <xsl:otherwise> para expresar múltiples pruebas condicionales.

SINTAXIS

```
<xsl:choose>
        <xsl:when test="expression">
                ... Hacer algo...
        </xsl:when>
        ...
        <xsl:otherwise>
                ... Hacer algo ....
        </xsl:otherwise>
</xsl:choose>
```

EJEMPLO

```
<xsl:for-each select="//ciudad">
<xsl:choose>
        <xsl:when test="habitants &lt; 100000">
                <trbgcolor="#ff0000">
                <td><xsl:value-of select="./nombre" /></td>
                <td><xsl:value-of select="./habitantes" /></td>
                </tr>
        </xsl:when>
        <xsl:otherwise>
                <trbgcolor="#00ff00">
                <td><xsl:value-of select="./nombre" /></td>
                <td><xsl:value-of select="./habitantes" /></td>
                </tr>
        </xsl:otherwise>
</xsl:choose>
</xsl:for-each>
```

xsl:apply-imports	**Invoca una regla de plantilla reemplazada.**
xsl:apply-templates	Dirige el procesador XSLT para que busque la plantilla adecuada que se debe aplicar, según el tipo y el contexto del nodo seleccionado.
xsl:attribute	Crea un nodo de atributo y lo asocia a un elemento resultante.
xsl:attribute-set	Define un conjunto de atributos con nombre.
xsl:call-template	Invoca una plantilla por nombre.
xsl:choose	Proporciona múltiples pruebas condicionales junto con los elementos `<xsl:otherwise>` y `<xsl:when>`.
xsl:comment	Genera un comentario en el resultado.
xsl:copy	Copia el nodo actual del origen al resultado.
xsl:copy-of	Inserta subárboles y fragmentos del árbol de resultados en el árbol de resultados.
xsl:decimal-format	Declara un formato-digital, que controla la interpretación de un modelo de formato utilizado por la función `format-number`.
xsl:element	Crea un elemento con el nombre especificado en el resultado.
xsl:fallback	Llama al contenido de la plantilla que puede proporcionar un sustituto razonable al comportamiento del nuevo elemento cuando se encuentre.

xsl:for-each	**Aplica una plantilla repetidamente, aplicándola a su vez en cada nodo de un conjunto.**
xsl:if	Permite obtener fragmentos de plantillas condicionales simples.
xsl:import	Importa otro archivo XSLT.
xsl:include	Incluye otro archivo XSLT.
xsl:key	Declara una clave para utilizar con la función `key()` en expresiones de lenguaje de ruta XML (XPath).
xsl:message	Envía un mensaje de texto al búfer del mensaje o al cuadro de diálogo del mensaje.
xsl:namespace-alias	Sustituye el prefijo relacionado con un espacio de nombres dado por otro prefijo.
xsl:number	Inserta un número con formato en el árbol de resultados.
xsl:otherwise	Proporciona múltiples pruebas condicionales junto con los elementos `<xsl:choose>` y `<xsl:when>`.
xsl:output	Especifica las opciones que se deben utilizar a la hora de serializar el árbol de resultados.
xsl:param	Declara un parámetro con nombre que se puede utilizar dentro de un elemento `<xsl:stylesheet>` o un elemento `<xsl:template>`. Permite especificar un valor predeterminado.
xsl:preserve-space	Conserva los espacios en blanco en un documento.
xsl:processing-instruction	Genera una instrucción de proceso en el resultado.

xsl:variable*	**Define variables y funciones globales para extensiones de secuencias de comandos.**
xsl:sort	Especifica los criterios de ordenación para las listas de nodos seleccionadas por `<xsl:for-each>` o `<xsl:apply-templates>`.
xsl:strip-space	Elimina espacios en blanco en un documento.
xsl:stylesheet	Especifica el elemento de documento en un archivo XSLT. El elemento de documento contiene otros elementos XSLT.
xsl:template	Define una plantilla reutilizable para generar los resultados deseados para los nodos de un tipo y contexto en particular.
xsl:text	Genera texto en el resultado.
xsl:transform	Lleva a cabo la misma función que `<xsl:stylesheet>`.
xsl:value-of	Inserta el valor del nodo seleccionado como texto.
xsl:variable	Especifica un valor enlazado de una expresión.
xsl:when	Proporciona múltiples pruebas condicionales junto con los elementos `<xsl:choose>` y `<xsl:otherwise>`.
xsl:with-param	Pasa un parámetro a una plantilla.