3. XSL (eXtensible Stylesheet Language = Lenguaje Extensible de Hojas de Estilo)

En el ejemplo del principio del tema, vimos como mostrar los datos de un documento XML, utilizando un documento XHTML.

En realidad, la idea que perseguimos es la siguiente:

- Crear un plantilla del documento que queremos obtener, en el ejemplo XHTML.
- En la plantilla indicar que datos queremos mostrar y en qué lugar.

Aunque no sea realmente así, supongamos que la plantilla que creamos para el documento XHTML que utiliza los datos del XML del ejemplo, sea la siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.1//EN"
        "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
        <head>
                <title>Libros</title>
        </head>
        <body>
                <h1>Libros</h1>
                <ul>
                        <li>****El título del un libro*****</li>
                        <li>****El título del otro libro*****</li>
                </ul>
        </body>
</html>
```

El proceso de transformación del XML al XHTML, utilizando una plantilla, sería:

Los documentos XSL serán los que contienen a las plantillas, necesarias para la transformación.

3.1. Estructura básica de un documento XSL.

Los documentos XSL mantienen la sintaxis XML, de manera que empezaremos declarando al documento XML. Seguidamente, añadimos el elemento raíz del documento XSL, que indica que se trata de este tipo de documento.

Sintaxis declarativa:

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

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

A continuación, indicaremos que vamos a trabajar con todo el documento XML para definir la plantilla, de manera que incluimos el elemento xsl:template dentro de xsl:stylesheet. Para ello, también asignamos a su atributo match el valor de la raíz XML (/).

```
<xsl:template match=”/”>

<xsl:template>
```

El documento XSL tundra el siguiente aspecto:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
                <!--Escribir el contenido de la plantilla-->
        </xsl:template>
</xsl:stylesheet>
```

3.2. Transformar de XML a XHTML.

Las transformaciones pueden generar archivos XML, XHTML y archivos de texto, por lo que es necesario indicar que tipo de transformación queremos hacer.

En el caso de que queramos hacer una trasformación de XML a XHTML, en la estructura básica del documento XSL, debemos **añadir una nueva línea** antes de xsl:template:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="xml" indent="yes"
                doctype-public="-//W3C//DTD XHTML 1.1/EN"
                doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"/>
        <xsl:template match="/">
                <!--Escribir el contenido de la plantilla-->
        </xsl:template>
</xsl:stylesheet>
```

Esta es la estructura principal para crear un documento XSL, destinado a transformar un documento XML en un documento XHTML.

```
<xsl:output
        method="xml|html|text|name"
        version="string"
        encoding="string"
        omit-xml-declaration="yes|no"
        standalone="yes|no"
        doctype-public="string"
```

```
        doctype-system="string"
        cdata-section-elements="namelist"
        indent="yes|no"
        media-type="string"
</xsl:output>
```

Attribute	Value	Description
method	xml html text	Opcional. Define el formato de salida. Por defecto, XML.
version	string	Opcional. Pone el número de vesrsión del W3C para el formato de salida (solo cuando el método es html o xml).
encoding	string	Opcional. Fija el valor de la codificación para el documento de salida.
omit-xml-declaration	yes no	Opcional. Por defecto, "no". "yes" indica que la declaración XML (<?xml…?>) puede ser omitida en el documento de salida.
standalone	yes no	Opcional.
doctype-public	string	Opcional. Fija el valor del atributo PUBLIC para el DOCTYPE del documento de salida.
doctype-system	string	Opcional. Fija el valor del atributo SYSTEM para el DOCTYPE del documento de salida (dtd).
cdata-section-elements	namelist	Opcional. Lista de elementos, separados con un espacio en blanco, cuyo contenido de texto puede ser escrito en las secciones CDATA.
indent	yes no	Opcional. "yes" indica que la salida puede estar sangrada según su estructura de texto.
media-type	string	Opcional. Por defecto, "text/html". Define el tipo MIME para el documento de salida.

EJEMPLO: Documento con tres tipos de salida (ejemplo libros outs).

3.3. Enlazar el documento XML y el documento XSL.

Para aplicar a un documento XML la trasformación definida en un documento XSL, tenemos que indicar al documento XML cuál es el documento XSL.

Sintaxis:

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

Ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="Ejemplo_apartado3_1.xsl"?>
<catalogo>
        <libro>
                <titulo>Manual imprescindible de C/C++</titulo>
                <isbn>9788441526143</isbn>
                <autores>
                        <autor>Miguel Angel</autor>
                </autores>
                <paginas>416</paginas>
                <editorial>Anaya Multimedia</editorial>
```

```
        </libro>
        <libro>
                <titulo>Redes locales</titulo>
                <isbn>9788441519800</isbn>
                <autores>
                        <autor>Jim Doherty</autor>
                        <autor>Neil Anderson</autor>
                </autores>
                <paginas>544</paginas>
                <editorial>Rama</editorial>
        </libro>
</catalogo>
```

Si ejecutamos este XML, veremos que aparece la ventana del navegador vacía, ya que aún no tenemos definida la plantilla.

> Nota:
> Chrome no realiza transformaciones XSL en modo local por motivos de seguridad. Solo funciona si la transformación se realiza de un documento alojado en un servidor de Internet. Luego, las transformaciones de nuestros ejemplos, en modo local, las haremos con el Explorer y el FireFox.

3.4. Crear la plantilla.

En el caso que nos ocupa, queremos obtener un documento XHTML, luego la plantilla deberá tener las siguientes líneas básicas:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
        <head>
                <title> </title>
        </head>
        <body>

        </body>
</html>
```

Entonces, el documento XSL con el que estábamos trabajando quedará así:

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/transform">
        <xsl:output method="xml" indent="yes"
                doctype-public="-//W3C//DTD XHTML 1.1//EN"
                doctype-system="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"/>
        <xsl:template match="/">
                <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
                        <head>
                                <title>Titulo de la pagina web</title>
                        </head>
                        <body>
```

```
                                    <h1>Libros</h1>
                                </body>
                            </html>
                        </xsl:template>
</xsl:stylesheet>
```

Como podemos ver la plantilla puede contener una página web completa, pero ¿Cómo indicamos en el documento XSL que queremos incluir datos del documento XML?

3.5. Seleccionar valores.

Vamos a ver cómo podemos seleccionar valores de un documento XML.

Elemento simple:

xsl: value – of

Sintaxis: <xsl:value-of select=”elementoXML” />

Para seleccionar el elementoXML debemos escribir la ruta Xpath.

Por ejemplo: seleccionar libros:

<xsl:value-of select=”/catalogo/libro/titulo” />

Iteración- Elemento compuesto:

<xsl:for-each select='*'>Elemento Compuesto<xsl:for-each>

Por ejemplo, para seleccionar todos los títulos de los libros:

```
<xsl:for-each select="/catalogo/libro">
        <h3><xsl:value-of select="titulo"/></h3>
</xsl:for-each>
```

3.6. Ordenar: xsl-sort

Se especifica dentro de xsl:apply-templates o xsl:for-each

- Su atributo es select
- Indica cómo se establece el orden.

Por ejemplo, para ordenar alfabéticamente todos los títulos de los libros:

```
<xsl:for-each select="/catalogo/libro">
        <xsl:sort select=”titulo”/>
        <h3><xsl:value-of select="titulo"/></h3>
</xsl:for-each>
```

3.7. Condicional:

xsl:if

Atributo: test

- El valor del atributo es una expresión booleana
- Las instrucciones que contiene se ejecutan sólo si la condición se cumple.

xsl:choose

Contiene elementos xsl:when

- Atributo: test (similar al de xsl:if)
- Son los diferentes "casos" de una sentencia CASE

Caso por defecto: xsl:otherwise (sin atributos)

```
<xsl:choose>
        <xsl:when test=" final>='9.0' ">Sobresaliente</xsl:when>
….
        <xsl:otherwise>Suspenso</xsl:otherwise>

</xsl:choose>
```

3.8. Resumen:

Elemento fundamental: xsl:stylesheet, en el que incluimos versión (1.0) y el *namespace* xsl

- Dentro de él, los elementos del nivel superior
- Utilizamos **xsl:output** para decir si la salida es XML, HTML o texto normal, y algunos detalles más de cómo se genera
- Utilizamos **xsl:template** como bloque básico
- Problemas (en este punto):
  - cómo escribir expresiones match para los templates
  - qué instrucciones podemos utilizar para generar la salida

xsl:apply-templates:

Se utiliza para indicar al procesador que intente emparejar templates con cierto nodo o conjunto de nodos (*nodeset*).

- Atributos:
  - select: Su valor es una expresión XPath de conjunto de nodos. El procesador intentará emparejar ese conjunto de nodos con sus templates respectivos
- xsl:apply-templates permite realizar un tratamiento recursivo de todos los elementos del árbol fuente

TUTORIAL con ejemplos:
http://www.w3schools.com/xsl/