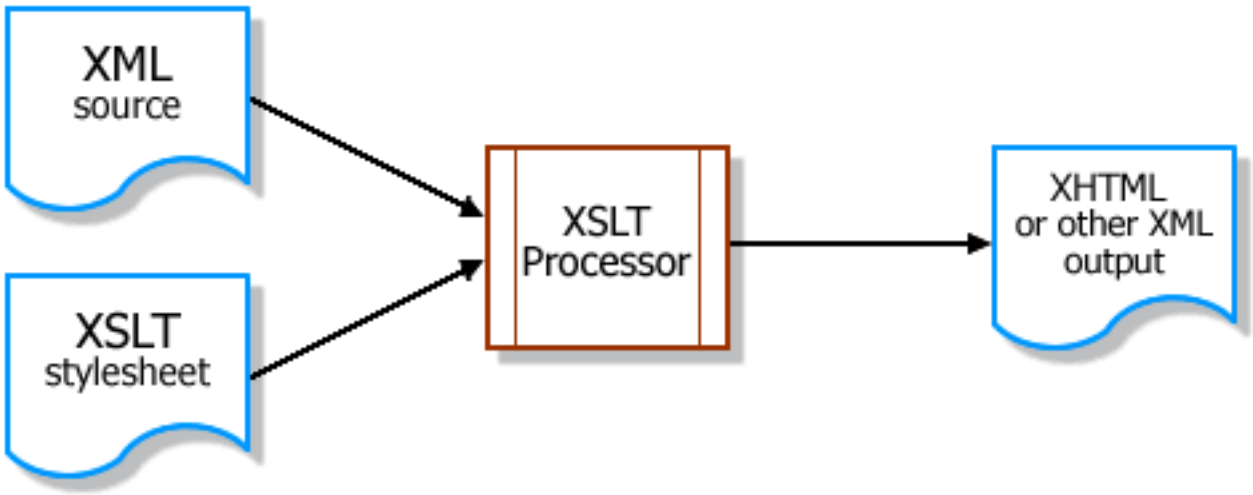
T4. Conversión de ficheros XML. XSLT



Introducción

Aunque las hojas de estilo CSS se pueden aplicar a los documentos XML, las hojas de estilo tienen muchas limitaciones ya que se crearon para complementar al HTML, es decir, para ver páginas web en pantallas de ordenador. De la misma manera que el **XML es una generalización del HTML**, el W3C creó una **generalización de las hojas de estilo CSS a la que se denominó XSL** (eXtensible Stylesheet Languaje), es decir, **Lenguaje de Hojas de Estilo Extensible**.

El W3C ha desarrollado tres lenguajes:

- **XPath**: un lenguaje para referirse a partes de un documento XML.
- **XSLT** (XSL Transformation): un lenguaje para transformar documentos XML.
- XSL-FO (XSL Formatting Objects, es decir, Objetos de formato XSL): un lenguaje para especificar el formato de un documento XML y posteriormente convertirlo a PDF o PostScript.

XSL y XSLT

¿Qué es XSL?

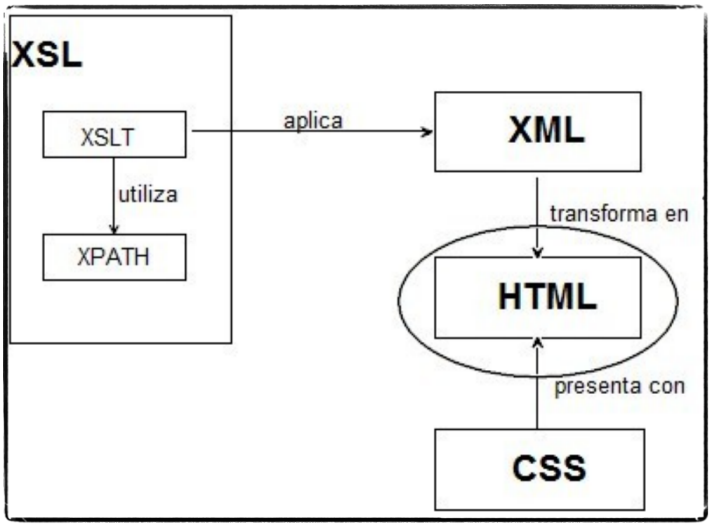
XSL (Extensible Stylesheet Language) que podemos traducir como lenguaje extensible de hojas de estilo, pero que en realidad es un **metalenguaje**, por estar formado por una familia de especificaciones o recomendaciones oficiales del W3C y que son XSLT, XPATH y XSL-FO.

Algunos autores opinan que **XSL es en XML** el equivalente del **CSS en HTML**. Por eso se le **llama hoja de estilo de XML** pero no podemos confundirlo, aunque tengan parte del nombre en común, y función similar basada en la presentación de datos XML en un formato legible.

Desde el principio del uso de XML hubo una gran preocupación por controlar la presentación de los documentos en la WEB y por esa razón se adaptó las tecnologías existentes como las CSS, y que nosotros ya hemos tratado también en temas anteriores. Pero aunque hemos podido comprobar su buen funcionamiento, CSS tiene unas limitaciones marcadas por la falta de construcciones de sentencias de control y filtros adecuados que lo hacen insuficiente cuando entramos de llenos en el mundo de la transformación de documentos.

Es una realidad que XSL permite definir hojas de estilo más adecuadas para los documentos XML frente a CSS que utiliza un método de presentación adaptado al mundo de HTML.

Pero eso **no quiere decir que XSL sustituya a las CSS**, ya que en la práctica XSL, HTML y CSS son complementarios, de forma que el proceso habitual consiste en que los datos de los documentos XML se utilizan en una página Web mediante su transformación a HTML con XSLT y la ayuda de XPATH, para finalmente presentar el documento HTML utilizando una CSS, como se representa en el siguiente gráfico:



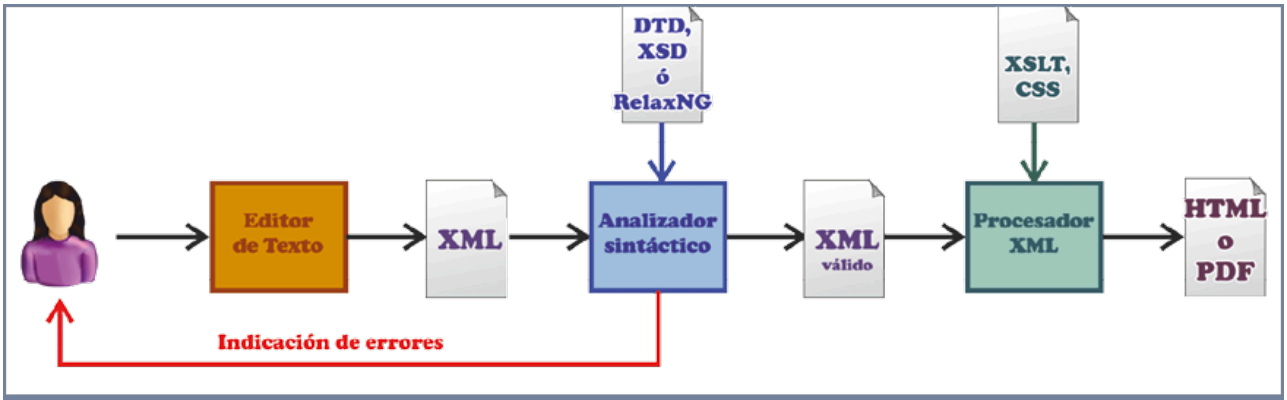


Ilustración 21. Proceso completo de funcionamiento productivo de un XML

¿Qué es XSLT?

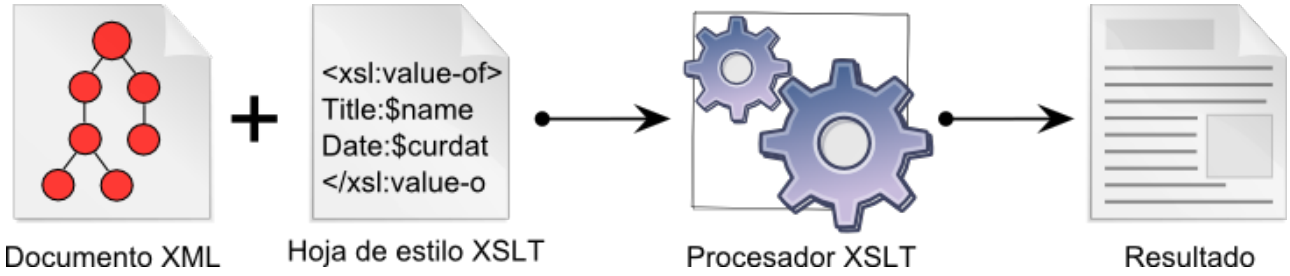
XSLT es la especificación concreta que dentro del metalenguaje XSL desarrolla el lenguaje de transformación. No obstante, y aclaradas las diferencias, entenderemos que ambos términos (XSL y XSLT) se usan en la práctica como equivalentes para referirse al propio proceso de transformación, siendo simplemente una distinción conceptual a tener en cuenta.

Para hacer las transformaciones se hace uso de la especificación XPath, que aunque en origen fue diseñada para ser utilizada de forma independiente tiene su completa utilidad embebido en una aplicación de transformación creando los filtros adecuados para obtener los elementos que queremos traspasar a los distintos documentos.

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**
- **Junio del 2017: XSLT 3.0**

XSLT (Transformaciones XSL) es un lenguaje de programación declarativo que permite generar documentos a partir de documentos XML (como ilustra la imagen). Los documentos generados pueden ser otros documentos XML o no.



- 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).

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.

Al ser XSLT un lenguaje declarativo las hojas de estilos no se escriben como una secuencia de instrucciones, sino como una colección de **plantillas** (**template rule**). Cada plantilla establece como se transforma un determinado elemento (definido con expresiones XPath).

Pasos para la transformación de un documento usando XSLT

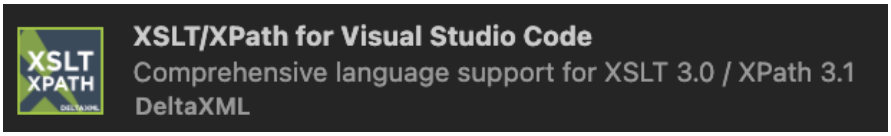
PASOS

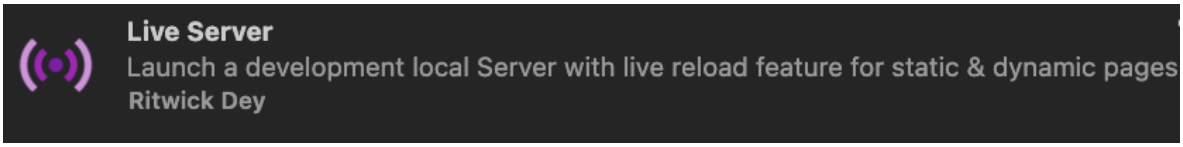
La **transformación del documento** se realiza de la siguiente forma:

- El procesador XSLT **analiza el documento y construye el árbol** del documento.
- El procesador recorre todos los nodos desde el nodo raíz, **aplicando a cada nodo una plantilla - template**, sustituyendo el nodo por el resultado.
- Cuando el procesador ha recorrido todos los nodos, se ha terminado la transformación.

Entorno de trabajo propuesto

Para la usar XSLT, podemos usar distintos entornos como son el XML Copy Editor, el Netbeans, el Visual Studio Code, etc. Yo les propongo usar el Visual Studio Code, en el cual deberemos de instalar los siguientes plugins para que nos facilite el trabajo.





Hola Mundo con XSLT

Para realizar el típico "Hola Mundo", lo vamos a realizar usando NetBeans y Visual Studio Code, para ver las peculiaridades de cada uno. Pero luego seguiremos trabajando con VSC.

Para poder transformarlo a HTML, simplemente tenemos que insertar entre el prólogo y el elemento raíz la instrucción de procesamiento **xml-stylesheet**:

<?xml-stylesheet href="holaMundo.xsl**" type="text/xsl"?>**

ciclosFormativos.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="holaMundo.xsl" type="text/xsl"?>
<ies nombre="IES Dos Barrancos" web="http://www.iesdosbarrancos.com/" >
  <ciclos>
    <ciclo id="ASIR">
      <nombre>Administración de Sistemas Informáticos en Red</nombre>
      <grado>Superior</grado>
      <decretoTitulo año="2009" />
    </ciclo>
    <ciclo id="DAW">
      <nombre>Desarrollo de Aplicaciones Web</nombre>
      <grado>Superior</grado>
      <decretoTitulo año="2010" />
    </ciclo>
    <ciclo id="SMR">
      <nombre>Sistemas Microinformáticos y Redes</nombre>
      <grado>Medio</grado>
      <decretoTitulo año="2008" />
    </ciclo>
  </ciclos>
</ies>
```

holaMundo.xsl

```xml
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>Hola Mundo</title>
        </head>
        <body>
            <h1>Hola Mundo</h1>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Aunque profundizaremos sobre la función de los distintos elementos del fichero XSL, veamos algunas cosas interesantes e imprescindibles para entender la sintaxis de la transformación:

- La **sintaxis que utiliza los ficheros xsl es sintaxis XML**, y debe respetar las restricciones que ya conocemos, como la de ser un documento bien formado.
- El **elemento raíz** será siempre **<xsl:stylesheet.....>** que contiene la versión que se está usando y la referencia al namespace declarado como xmlns:xsl="http://www.w3.org/1999/XSL/Transform" por lo que se usará, en el resto de etiquetas "propias" el prefijo "xsl:"
- La etiqueta **<xsl:template match='/'>** hace referencia al elemento template cuyo atributo match especifica los nodos del documento origen que se utilizarán (usando para ello una expresión XPath), por lo que en este caso se refiere al elemento raíz y abarcará todo el documento, que será luego empleado al invocarlo mediante la orden <xsl:apply-templates />.
- En este caso, la salida ha sido un documento html, <**xsl:output** method="html" indent="yes"/>.

AC1.1

Realiza el ejercicio hola mundo usando Netbeans y Visual Studio Code.

Tipos de archivo de destino.

Las transformaciones XSL pueden obtener archivos de tipo xml, txt, pdf y **html**. Se puede especificar el método a utilizar si incluimos el elemento **xsl:output** con su atributo **method**. Cuando no se especifica el método de salida, se sigue la siguiente norma:

- El método de salida por defecto cuando no se especifica ningún método, será xml.
- No obstante cuando el primer elemento que se especifica (que no sea directamente un elemento xsl, es decir, que no lleva el prefijo xsl:) sea <html> el documento resultante será html.

Otro atributo optativo del elemento output es **indent** que puede contener los valores "yes" o "no". Su valor por defecto es "no", por lo que si queremos que la salida se muestre con cada elemento en una línea y los elementos hijos identandos automáticamente, tendremos que poner su valor a "yes".

<xsl:output method="html" indent="yes" />

Enlace donde puedes ver los atributos que puede tomar y sus respectivos valores:

xsl:output —> https://www.w3schools.com/xml/ref_xsl_el_output.asp

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

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

...

...

</xsl:stylesheet>
```

```
<?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" version="1.0"
encoding="UTF-8" indent="yes"/>

...

...

</xsl:stylesheet>
```

Elementos XSLT

Existen muchos elementos pertenecientes a la tecnología XSLT de los cuales solo veremos una selección de los mismos, correspondientes a los mas utilizados.

Listado de elementos XSLT:

- **https://www.w3schools.com/xml/xsl_elementref.asp**
- **https://www.data2type.de/es/xml-xslt-xslfo/xslt/referencia-xslt1**

Template.

Una plantilla es un bloque utilizado con el elemento que delimita una serie de contenidos y reglas XSL como centro de cualquier transformación. Su importancia dentro de XSLT es de primer nivel, pues **la transformación consistirá simplemente en aplicar una colección de esas plantillas al documento de entrada para obtener el correspondiente documento de salida**. El **atributo match** asociará la plantilla con un elemento XML, usando una expresión XPath.

<xsl:template match="expresión XPath">

La plantilla esta formada por contenidos que queremos incluir en nuestro documento de salida, como las distintas etiquetas html y/o por instrucciones propias de xslt, que son las que comenzarán con el prefijo declarado en namespace y que en este caso es **' xsl: '**.

Es decir, todo lo que está dentro de la plantilla formará la salida o resultado, trasladando todos los elementos normales o texto tal cual están, mientras que **los que tienen el prefijo 'xsl:' indicarán al procesador que se deberá hacer alguna cosa con ellos**.

Como vimos en el ejemplo del Hola Mundo.

IMPORTANTE: Cuando se aplica una plantilla a un nodo, **en principio no se recorren los nodos descendientes.** Para indicar que sí queremos recorrer los nodos descendientes y aplicarles las plantillas que les correspondan, hay que utilizar la instrucción **<xsl:apply-templates />**,

Value-of

Mediante el elemento **value-of** y usando el atributo **select** podemos utilizar el valor que se encuentra en cada nodo. El valor del **atributo select** es una expresión XPath. La expresión XPath utilizada puede servir para devolver tanto el valor del texto asociado al nodo como el del atributo correspondiente (recordemos que el **atributo deber ir precedido de @**. Su sintaxis es:

<xsl:value-of select="expresión XPath">

Text

El elemento **<xsl:text> inserta texto**, aunque no es necesario ya que en las plantillas podemos escribir directamente el texto que queremos que se escriba en el árbol de resultado. Ambos ejemplos tendrían el mismo resultado final.

```
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>Mostar Elementos</title>
        </head>
        <body>
            <h1>Mostrar Elementos y Atributos</h1>
            <p>
                <xsl:value-of select="//ciclo[1]/nombre"/>
                <xsl:text> (</xsl:text>
                <xsl:value-of select="//ciclo[1]/decretoTitulo/@año"/>
                <xsl:text>)</xsl:text>
            </p>
            <p>
                <xsl:value-of select="//ciclo[2]/nombre"/>
                <xsl:text> (</xsl:text>
                <xsl:value-of select="//ciclo[2]/decretoTitulo/@año"/>
                <xsl:text>)</xsl:text>
            </p>
            <p>
                <xsl:value-of select="//ciclo[3]/nombre"/>
                <xsl:text> (</xsl:text>
                <xsl:value-of select="//ciclo[3]/decretoTitulo/@año"/>
                <xsl:text>)</xsl:text>
            </p>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

AC1.2

Realiza un fichero xsl que extraiga la información del nombre de los ciclos y el año de publicación del decreto que lo regula.

Mostrar Elementos y Atributos

Administración de Sistemas Informáticos en Red (2009)

Desarrollo de Aplicaciones Web (2010)

Sistemas Microinformáticos y Redes (2008)

Apply-templates

El elemento **<xsl:apply-templates>** se utiliza desde dentro de una plantilla **para llamar a otras**, y si no hay otras se aplica la plantilla por defecto que lo único que hace es incluir el contenido de las etiquetas en el documento de salida. Su sintaxis básica es:

```
<xsl:apply-templates select="expresión XPath">
```

El **atributo select**, es opcional, y 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.

```
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>Apply-Templates</title>
        </head>
        <body>
            <h1>Apply-Templates</h1>
            <xsl:apply-templates select="//ciclo"/>
        </body>
        </html>
    </xsl:template>
    <xsl:template match="ciclo">
        <p>
            <xsl:value-of select="nombre"/>
            <xsl:text> (</xsl:text>
            <xsl:value-of select="decretoTitulo/@año"/>
            <xsl:text>)</xsl:text>
        </p>
    </xsl:template>
</xsl:stylesheet>
```

AC1.3

Realiza un fichero xsl que extraiga la información del nombre de los ciclos y el año de publicación del decreto que lo regula, usando la apply-templates.

Apply-Templates

Administración de Sistemas Informáticos en Red (2009)

Desarrollo de Aplicaciones Web (2010)

Sistemas Microinformáticos y Redes (2008)

Apply-templates. Atributo mode.

El atributo **mode** se utiliza para definir múltiples plantillas con el mismo patrón de coincidencia, pero con **diferentes modos de procesamiento**. Esto permite crear varias transformaciones para el mismo elemento en un documento XML, cada una con un comportamiento diferente basado en el modo especificado. Veamos un ejemplo:

```xml
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>Apply-Templates.Mode</title>
        </head>
        <body>
            <h1>Apply-Templates.Mode</h1>
            <h2>Muestra Lista</h2>
            <ul>
                <xsl:apply-templates select="//ciclo" mode="lista"/>
            </ul>

            <h2>Muestra Tabla</h2>
            <table>
                <tr>
                    <th>Titulo</th>
                    <th>Año</th>
                </tr>
                <xsl:apply-templates select="//ciclo" mode="tabla"/>
            </table>
        </body>
        </html>
    </xsl:template>
    <xsl:template match="ciclo" mode="lista">
        <li>
            <xsl:value-of select="nombre"/>
            <xsl:text> (</xsl:text>
            <xsl:value-of select="decretoTitulo/@año"/>
            <xsl:text>)</xsl:text>
        </li>
    </xsl:template>
    <xsl:template match="ciclo" mode="tabla">
        <tr>
            <td><xsl:value-of select="nombre"/></td>
            <td><xsl:value-of select="decretoTitulo/@año"/></td>
        </tr>
    </xsl:template>
</xsl:stylesheet>
```

Apply-Templates.Mode

Muestra Lista

- Administración de Sistemas Informáticos en Red (2009)
- Desarrollo de Aplicaciones Web (2010)
- Sistemas Microinformáticos y Redes (2008)

Muestra Tabla

Titulo	Año
Administración de Sistemas Informáticos en Red	2009
Desarrollo de Aplicaciones Web	2010
Sistemas Microinformáticos y Redes	2008

AC1.4

Realiza un fichero xsl que extraiga la información del nombre de los ciclos y el año de publicación del decreto que lo regula y que nos lo muestre en formato de lista y tabla. Aplica una hoja de estilos CSS para que se muestre los bordes y colores.

Sort

Se utiliza dentro de un <xsl:apply-templates> o un <xsl:foreach> para ordenar según lo que le indiquemos en el atributo **select** y de forma ascendente o descendente según le indiquemos en el atributo **order**.

```
<xsl:sort select="expresión XPath" order="ascending | descending" "/>
```

```xml
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>Sort</title>
        </head>
        <body>
            <h1>Sort</h1>
            <xsl:apply-templates select="//ciclo">
                <xsl:sort select="nombre" order="descending"/> <!-- ascending -->
            </xsl:apply-templates>
        </body>
        </html>
    </xsl:template>
    <xsl:template match="ciclo">
        <p>
            <xsl:value-of select="nombre"/>
            <xsl:text> (</xsl:text>
            <xsl:value-of select="decretoTitulo/@año"/>
            <xsl:text>)</xsl:text>
        </p>
    </xsl:template>
</xsl:stylesheet>
```

Sort

Sistemas Microinformáticos y Redes (2008)

Desarrollo de Aplicaciones Web (2010)

Administración de Sistemas Informáticos en Red (2009)

AC1.5

Realiza un fichero xsl que extraiga la información del nombre de los ciclos y el año de publicación del decreto que lo regula, ordenado descendentemente por el año de publicación y en formato de tabla.

Elementos de Control

En XSLT se pueden usar filtros y estructuras que facilitan las operaciones de control del contenido de los documentos. Estas operaciones de control son habituales en los lenguajes de programación tradicionales, y se conocen como:

- **Iteraciones** (o repeticiones o bucles). **for-each**
- **Selecciones** (o alternativas o condicionales). **if-test y choose**

Iteraciones: for-each

Este elemento se usa para repetir la búsqueda de los nodos que coinciden con la expresión XPath que se usa en el atributo select, de forma que solo escribimos una vez el código que comprende la instrucción, pero **se aplica a todos los casos en que se cumpla la expresión**. Su sintaxis es:

```
<xsl:for-each select="expresión XPath" />
```

```
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>For-Each</title>
        </head>
        <body>
            <h1>For-Each</h1>
            <xsl:for-each select="//ciclo">
                <xsl:sort select="decretoTitulo/@año" order="descending"/>
                <p>
                    <xsl:value-of select="nombre"/>
                    <xsl:text> (</xsl:text>
                    <xsl:value-of select="decretoTitulo/@año"/>
                    <xsl:text>)</xsl:text>
                </p>
            </xsl:for-each>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

For-Each

Desarrollo de Aplicaciones Web (2010)

Administración de Sistemas Informáticos en Red (2009)

Sistemas Microinformáticos y Redes (2008)

AC1.6

Realiza un fichero xsl que extraiga la información del nombre de los ciclos y el año de publicación del decreto que lo regula, usando for-each. Debe mostrarlo ordenado descendentemente según el año.

Selección: if test

Disponemos también de elementos que nos permiten decidir si una acción se realizará o no dependiendo de ciertos criterios, es decir, generar contenido condicionalmente. Su sintaxis sería:

```
<xsl:if test="expresión XPath" >
        …..
</xsl:if>
```

El **atributo test** es obligatorio y contiene, en sintaxis XPath, la condición que se tiene que cumplir. Veamos el siguiente ejemplo en el que se muestra los ciclos en los que se publico su real decreto posterior a 2009, está realizado con if y también se puede hacen con for-each:

```xml
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>IF</title>
        </head>
        <body>
            <h1>IF</h1>
            <xsl:for-each select="//ciclo">
                <xsl:sort select="decretoTitulo/@año" order="descending"/>
                <xsl:if test="decretoTitulo[@año &gt; 2009]">
                    <p>
                        <xsl:value-of select="nombre"/>
                        <xsl:text> (</xsl:text>
                        <xsl:value-of select="decretoTitulo/@año"/>
                        <xsl:text>)</xsl:text>
                    </p>
                </xsl:if>
            </xsl:for-each>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

```xml
<?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" indent="yes"/>
    <xsl:template match="/">
        <html lang="es">
        <head>
            <meta charset="UTF-8"/>
            <title>IF</title>
        </head>
        <body>
            <h1>IF</h1>
            <xsl:for-each select="//ciclo[decretoTitulo/@año &gt; 2009]">
                <xsl:sort select="decretoTitulo/@año" order="descending"/>
                <p>
                    <xsl:value-of select="nombre"/>
                    <xsl:text> (</xsl:text>
                    <xsl:value-of select="decretoTitulo/@año"/>
                    <xsl:text>)</xsl:text>
                </p>
            </xsl:for-each>
        </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

IF

Desarrollo de Aplicaciones Multiplataforma (2011)

Desarrollo de Aplicaciones Web (2010)

AC1.7

Realiza un fichero xsl que extraiga la información de los ciclos que el año de publicación del decreto que lo regula sea posterior a 2009.

AC1.8

Realiza un fichero xsl que extraiga la información del ciclo que tiene el currículo más actual y lo muestre en formato tabla.

AC1.9

Realiza un fichero xsl que extraiga la información de los tres ciclos que tiene el currículo más actual y lo muestre en formato tabla.

Otro ejemplo sencillo, restringiendo la búsqueda a un determinado nombre:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet                                                    version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match='/'>
      <html>
      <head>
          <title>Ejemplo8</title></head>
       <body>
           <xsl:for-each select="grupo/nombre">
               <xsl:if test=".='Carmen' ">
                    <p>He encontrado a <xsl:value-of select="." /> </p>
               </xsl:if>
           </xsl:for-each>
      </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

He encontrado a Carmen

NOTA: Se han tenido que alternar las comillas ' y " para abarcar tanto las correspondientes a la expresión propia de la condición de test como a la del XPath que contiene, ya que al anidar unas comillas dentro de otras, deben ser distintas.

Enlaces de interés

En Inglés:

- La W3School —> https://www.w3schools.com/xml/xpath_intro.asp
- La W3C —> https://www.w3.org/TR/xpath/
- Quick Reference —> http://geneura.ugr.es/~victor/cursillos/xml/XPath/XSLTquickref.pdf

En Español:

- Traducción del W3C —> http://www.sidar.org/recur/desdi/traduc/es/xml/xpath.html
- Con ejemplos XPath —> http://geneura.ugr.es/~victor/cursillos/xml/XPath/index.html
- Explicación XPath —> http://www.mclibre.org/consultar/xml/lecciones/xml_xpath.html
- Explicación XSLT —> http://www.mclibre.org/consultar/xml/lecciones/xml_xslt.html