UT4. CONVERSIÓN Y ADAPTACIÓN DE DOCUMENTOS XML

1º CURSO CFGS DAW

LMSGI

Contenido

1. Introducción

Las transformaciones XSL nos permiten modificar el contenido de un documento XML a nuestro antojo. Podemos pasar XML a un formato totalmente diferente, pero conservando los datos del original. De esa forma, lo que en origen era un documento XML puede convertirse en un archivo de texto, con sus datos separados por comas, en una página HTML, o en cualquier formato que sea necesario.

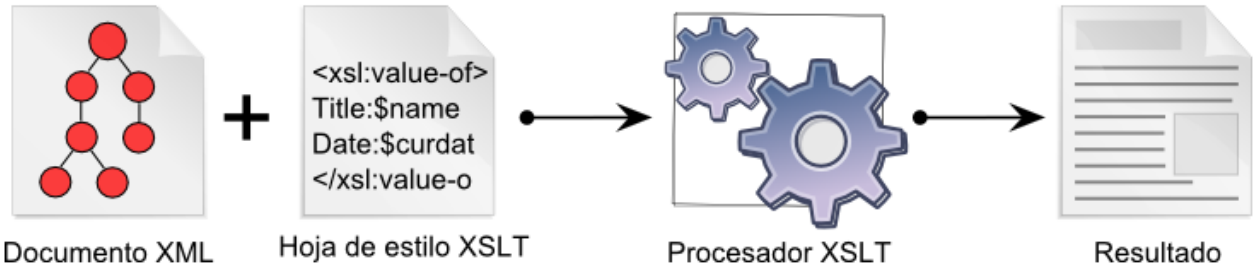
Es decir, XSLT (eXtensible Stylesheet Language-Transformations; en español, Lenguaje de hoja de estilo ampliable para transformaciones) describe un lenguaje, basado en XML, para transformar documentos XML a cualquier otro formato.

Durante este capítulo veremos cómo realizar estas transformaciones, en las que podemos ser todo lo selectivos que queramos con los datos que queremos mostrar.

Si el documento XML contiene información sobre películas, directores y actores, podemos limitarnos a mostrar sólo el título de las películas.

Una de las ventajas de estas transformaciones está en que son reutilizables. Si funcionan con documentos XML que sigue un determinado patrón, funcionarán con todos los que sigan dicho formato.

El paso de XML a HTML es el más frecuentemente utilizado, por lo vistoso que resulta. Además, si en un momento dado queremos cambiar la distribución de los elementos dentro de la página HTML sólo será necesario crear una nueva transformación, manteniéndose los datos del documento XML intactos.



Aunque existen herramientas para realizar las transformaciones (XSLT), como XMLspy, seguiremos utilizando la herramienta gratuita XML Copy Editor y un navegador para observar el resultado HTML de dichas transformaciones.

La transformación del documento se realiza la siguiente manera.

- El procesador analiza el documento y construye el árbol del documento.
- El procesador recorre el árbol del documento desde el nodo raíz.
- En cada **nodo recorrido**, el procesador aplica o no alguna plantilla:

- Si a un nodo no se le puede aplicar ninguna plantilla se incluye en el documento final (el texto del nodo). A continuación se recorren los nodos hijos.
- Si a un nodo se le puede aplicar una plantilla, se aplica la plantilla. La plantilla puede generar texto adicional.

- Cuando el procesador ha recorrido el árbol, se ha terminado la transformación.

2. Estructura de una hoja de estilo XSLT

- Una hoja de estilo XSLT es un documento XML que debe estar bien formado.
- Las hojas de estilo se guardarán siempre en archivos independientes con extensión **.xsl**
- Deben comenzar con una declaración XML. Es decir, <?xml version="1.0" ?>
- El elemento raíz de la hoja de estilo XSL 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="UTF-8"?>
<xsl:stylesheet xmlns:xsl= "http://www.w3.org/1999/XSL/Transform" version="1.0">

… Reglas de transformación - PLANTILLAS
</xsl:stylesheet>
```

- Entre las marcas de inicio y de fin del elemento raíz (**xsl:stylesheet**), se escribirán las **reglas de transformación** que se llamarán **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**. En este caso, el valor del atributo match es una expresión XPath. En concreto, **match="/"** que define todo el documento.
- La regla indica qué elementos del documento se van a transformar.

3. Enlazar documentos XML con hojas XSLT

Se puede asociar de forma permanente una hoja de estilo XSLT a un documento XML mediante la instrucción de procesamiento **<?xml-stylesheet ?>**, la misma que permite asociar hojas de estilo CSS. La instrucción de procesamiento **<?xml-stylesheet ... ?>** va al principio del documento, después de la declaración XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="prueba.xsl" ?>
```

Cuando se visualiza en un navegador web un documento XML enlazado con una hoja de estilo XSLT, los navegadores muestran el resultado de la transformación, aunque si se muestra el código fuente de la página, los navegadores muestran el documento XML original.

Lo dicho en el párrafo anterior es una verdad a medias. En algunos navegadores no se muestra directamente y necesitan que se ejecute en un servidor “https”, como en Chrome.
Por tanto, utilizaremos el propio editor XML Copy Editor para ver la transformación o lo abriremos directamente con el navegador Firefox.

4. Probando plantillas XLT

Nada mejor que empezar probando ejemplos de plantillas. Lo haremos sobre el siguiente documento:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="prueba.xsl" ?>

<biblioteca>
        <libro>
                <titulo>Don Quijote de la Mancha</titulo>
                <autor nacimiento="1547">Miguel de Cervantes</autor>
                <publicacion fecha="1605"/>
        </libro>

        <libro>
                <titulo>La colmena</titulo>
                <autor nacimiento="1916">Camilo José Cela</autor>
                <publicacion fecha="1951"/>
        </libro>

        <libro>
                <titulo>El lazarillo de Tormes</titulo>
                <autor> Anónimo</autor>
                <publicacion fecha="1554"/>
        </libro>
</biblioteca>
```

PLANTILLAS VACÍAS O NO EXISTENTES

Ejemplo 1 de 3

Si no hay plantillas, el procesador simplemente recorre todos los nodos y extrae el texto contenido por cada nodo.

En el ejemplo siguiente, el resultado incluye el contenido de los nodos <título> y <autor> puesto que no hay ninguna plantilla.

XSLT:	**Resultado**
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Don Quijote de la Mancha Miguel de Cervantes La colmena Camilo José Cela El lazarillo de Tormes Anónimo

Ejemplo 2 de 3

En el ejemplo siguiente, el resultado incluye el contenido de los nodos <titulo>, ya que no hay regla para ellos, pero los de <autor> se pierden porque **la plantilla es vacía**.

XSLT:	Resultado
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> **<xsl:template match="//autor"> </xsl:template>** </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Don Quijote de la Mancha La colmena El lazarillo de Tormes

Ejemplo 3 de 3

En el caso más extremo, **si la plantilla vacía se aplica al nodo raíz**, el procesador no genera ningún resultado en el documento final ni recorre ningún nodo hijo.

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

5. La instrucción <xsl:value-of>

La instrucción **<xsl:value-of>** extrae el contenido del nodo seleccionado.

En el ejemplo siguiente, el documento final contiene los autores de los libros porque la plantilla los genera con la instrucción <xsl:value-of>. Como se ha aplicado una plantilla al nodo **<libro>** y luego se selecciona **<autor>**, el resto de hijos (<titulo> y <publicacion>) no se recorren.

XSLT:	Resultado
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:template **match="//libro"**> **<xsl:value-of select="autor"**/> </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Miguel de Cervantes Camilo José Cela Anónimo

Observación. Aunque es algo que trataremos en otro capítulo (XPath), la expresión **//libro** indica que debemos posicionarnos en ese elemento, esté en el lugar que esté. Otra opción hubiera sido indicar toda su ruta. Es decir, /biblioteca/libro/autor. Después, al poner **"autor"**, se indica que se trata de un elemento hijo del elemento posicionado.

En el ejemplo siguiente, se obtienen el **título** y el **autor** de los libros, pero **uno a continuación de otro**. **Los saltos de línea se crean tras cada aplicación de la regla**.

XSLT:	Resultado
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:template **match="//libro"**> **<xsl:value-of select="titulo"/>** **<xsl:value-of select="autor"/>** </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Don Quijote de la ManchaMiguel de Cervantes La colmenaCamilo José Cela El lazarillo de TormesAnónimo

Para **extraer los atributos** es preciso utilizar otra expresión (también XPath) con el carácter **"@"**.

En el ejemplo siguiente, **las fechas de publicación se obtienen gracias a la regla que extraen el valor del atributo** y **los títulos y autores se obtienen porque al no haber reglas para ese nodo se extrae el contenido**.

XSLT:	Resultado
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:template **match="//publicacion"**> <xsl:value-of **select="@fecha"/**> </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Don Quijote de la Mancha Miguel de Cervantes 1605 La colmena Camilo José Cela 1951 El lazarillo de Tormes Anónimo 1554

Pero claro, ahora surge una duda, **¿cómo extraer sólo las fechas de publicación?** Con lo que sabemos hasta ahora podemos aplicar el siguiente razonamiento: primero indicaremos que la regla afectará al nodo **"libro"** y

después expresamos que dentro de él sólo queremos la fecha de publicación. Observa:

XSLT:	**Resultado**
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:template **match="//libro"**> <xsl:value-of **select="publicacion/@fecha"**/> </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> 1605 1951 1554

6. Generar texto adicional

Se puede generar texto escribiéndolo en la regla, por ejemplo, código html.

En el ejemplo siguiente se obtienen los **nombres** de los autores porque la regla selecciona el nodo **<libro>**, pero además generamos las etiquetas . El resultado permite intuir que la primera transformación HTML está más cerca.

XSLT:	Resultado
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:template **match="//libro"**> ** **<xsl:value-of **select="autor"**/>** ** </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Miguel de Cervantes Camilo José Cela Anónimo

Dentro de la regla podemos hacer referencia a varios subnodos. **En el siguiente ejemplo** se obtienen los nombres de los **autores** y los **títulos** de los libros porque la regla selecciona el nodo **<libro>** como en el ejemplo anterior, pero además generamos las etiquetas .

XSLT:	Resultado
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:template **match="//libro"**> ** **<xsl:value-of **select="autor"**/>** ** ** **<xsl:value-of **select="titulo"**/>** ** </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Miguel de Cervantes Don Quijote de la Mancha Camilo José Cela La colmena Anónimo El lazarillo de Tormes

Una curiosidad. Los siguientes ejemplos intentan conseguir el mismo resultado que el ejemplo anterior **utilizando dos reglas**, y no lo consiguen. **El procesador XSLT sólo aplica una regla a cada nodo**, la última, en este caso.

XSLT:	Resultado
<xsl:template match="//libro"> <xsl:value-of select="**autor**"/> </xsl:template> <xsl:template match="//libro"> <xsl:value-of select="**titulo**"/> </xsl:template>	<?xml version="1.0" encoding="UTF-8"?> Don Quijote de la Mancha La colmena El lazarillo de Tormes
XSLT	**Resultado**

<xsl:template match="//libro"> <xsl:value-of select="**titulo**"/> </xsl:template> <xsl:template match="/libro"> <xsl:value-of select="**autor**"/> </xsl:template>	<?xml version="1.0" encoding="UTF-8"?> Miguel de Cervantes Camilo José Cela Anónimo

Además de generar etiquetas, se puede generar texto. En el ejemplo siguiente se generan frases a partir del contenido de los nodos.

XSLT:	**Resultado**
<?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"> <xsl:template match="libro"> ** <xsl:value-of select="autor"/> escribió "<xsl:value-of select="titulo"/>" ** </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> Miguel de Cervantes escribió "Don Quijote de la Mancha" Camilo José Cela escribió "La colmena" Anónimo escribió "El lazarillo de Tormes"

Haz lo que consideres necesario para mostrar en el texto, el año de nacimiento del autor: por ejemplo – Miguel de Cervantes, quién nació en el año 1547, escribió “Don Quijote de la Mancha”

7. Aplicar reglas a subnodos: <xsl:apply-templates>

Sirve para llamar a otras plantillas desde el interior de una plantilla.

En el ejemplo siguiente, se genera la etiqueta **<html>** además de unos párrafos con los nombres de los **libros**. Este ejemplo sí que se puede ver en el navegador ya que se interpreta como html. ¡Nuestra primera transformación de verdad XML-HTML!

XSLT:	**Resultado**
<?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>Ejemplo XSLT</title> </head> <body> Libros de la biblioteca <**xsl:apply-templates** /> </body> </html> </xsl:template> <**xsl:template match="//libro"**> <xsl:value-of select="**titulo**" /> </xsl:template> </xsl:stylesheet>	<?xml version="1.0" encoding="UTF-8"?> <html> <head> <title>Ejemplo XSLT</title> </head> **<body> ** Libros de la biblioteca Don Quijote de la Mancha La colmena El lazarillo de Tormes </body> </html>

Hemos comenzado diciendo en este nuevo apartado:

> "La instrucción **<xsl:apply-templates>** hace que **se apliquen a los subelementos las reglas que les sean aplicables**."

Pues bien, para entender mejor las posibilidades de uso de esta nueva instrucción analiza, y prueba después, el siguiente código. Presta especial atención a aquello que aparece en negrita.

XSLT:

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

<xsl:template match="/">
<html>
        <head>
                <title>Ejemplo XSLT</title>
        </head>
        <body>
                <h1> Libros de la biblioteca </h1>

                <h2>Títulos de libros</h2>
                <xsl:apply-templates
                select="//titulo"/>

                <h2>Autores de libros</h2>
                <xsl:apply-templates
                select="//autor"/>

        </body>
</html>
</xsl:template>

<xsl:template match="//titulo">
<h3> <xsl:value-of select="."/> </h3>
</xsl:template>

<xsl:template match="//autor">
<h3> <xsl:value-of select="."/> </h3>
</xsl:template>

</xsl:stylesheet>
```

Resultado

```
<?xml version="1.0"
encoding="UTF-8"?>
<html>
  <head>
    <title>Ejemplo XSLT</title>
  </head>
  <body>
    <h1> Libros de la biblioteca
</h1>
    <h2>Títulos de libros</h2>
    <h3>Don Quijote de la
Mancha</h3>
    <h3>La colmena</h3>
    <h3>El lazarillo de
Tormes</h3>
    <h2>Autores de
libros</h2>
    <h3>Miguel de
Cervantes</h3>
    <h3>Camilo José Cela</h3>
    <h3>Anónimo</h3>
  </body>
</html>
```

Analicemos **<xsl:apply-templates select="//titulo"/>**. Se indica que se aplique la regla al elemento hijo **titulo**. En esta plantilla aparece una nueva expresión XPath: **select="."**. Esta expresión hace referencia al contenido del nodo actual

Haz cuantas pruebas consideres sobre el código anterior.

8. Saltos de línea y espacios en blanco

Al transformar un documento, los procesadores XSLT incorporan saltos de línea y espacios en blanco en el resultado, pero no lo hacen de forma uniforme.
No parece haber una solución sencilla que funcione en todos los procesadores, pero sí soluciones que funcionen en cada uno de ellos.

LA INSTRUCCIÓN <xsl:strip-space>

En el caso de XML Copy Editor, la forma más sencilla de mejorar el formato de presentación de los resultados, eliminando líneas en blanco innecesarias y sangrando los elementos anidados, es utilizar la instrucción <xsl:strip-space>. Pero debe tenerse en cuenta que esta instrucción no produce el mismo resultado en otros procesadores XSLT.

La instrucción <xsl:strip-space> permite indicar si los elementos que contienen únicamente espacios en blanco se incluyen en la transformación o no.

En el último ejemplo de la página siete, la etiqueta **<h1>** se generaba en la misma línea que la etiqueta **<body>**, pero en el ejemplo siguiente se generan en líneas distintas (y las etiquetas se muestran sangradas) al utilizar la instrucción **<xsl:strip-space>**:

XSLT:

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

<xsl:strip-space elements="*" />
<xsl:template match="/">
        <html>
                <head>
                        <title>Ejemplo XSLT</title>
                </head>
                <body>
                        <h1> Libros de la biblioteca </h1>
                        <xsl:apply-templates />
                </body>
                </html>
        </xsl:template>

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

Resultado

```
<?xml version="1.0"
encoding="UTF-8"?>
<html>
  <head>
    <title>Ejemplo XSLT</title>
  </head>
  <body>
    <h1> Libros de la biblioteca
</h1>
    <h3>Don Quijote de la
Mancha</h3>
    <h3>La colmena</h3>
    <h3>El lazarillo de
Tormes</h3>
  </body>
</html>
```

LA INSTRUCCIÓN <xsl:text>

Otra forma de mejorar la presentación mediante la instrucción **<xsl:text>**. **Pero esta instrucción no permite eliminar líneas en blanco**.
En el siguiente ejemplo, la etiqueta **<h1>** y **<body>** aparecen en líneas distintas al añadir un carácter de **salto de línea** (
) **y un par de espacios** para alinear la etiqueta **<h1>** con **<body>**.

```
<body>
<xsl:text>&#10;  </xsl:text>
<h1> Libros de la biblioteca </h1>
<xsl:apply-templates />
</body>
```

9. La instrucción <xsl:attribute>

La instrucción <xsl:attribute> permite generar un atributo y su valor. **Se utiliza cuando el valor del atributo se obtiene a su vez de algún nodo**.

A partir del documento XML, **se quiere generar la etiqueta <img>**. En la que el valor del atributo **src** seráa el contenido de la etiqueta **<fotografia>**.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="prueba.xsl" ?>

<libro>
        <titulo>Don Quijote de la Mancha</titulo>
        <autor>Miguel de Cervantes</autor>
        <fotografia>cervantes.png</fotografia>
</libro>
```

PRIMERA TRANSFORMACIÓN BÁSICA

XSLT:

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

<xsl:template match="/libro">
        <p>
          <img>
            <xsl:attribute name="src">
                <xsl:value-of select="fotografia"
        />
            </xsl:attribute>
          </img>
        </p>
</xsl:template>

</xsl:stylesheet>
```

Resultado

```
<?xml version="1.0"
encoding="UTF-8"?>
<p>
    <img
src="cervantes.png"/>
</p>
```

En la hoja de estilo XSLT, **la etiqueta <img> se escribe con apertura y cierre**, pero en el **resultado aparece como etiqueta única**.

VERSIÓN EN FORMATO HTML

XSLT:

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

 <xsl:template match="/">
  <html>
   <body>
    <xsl:apply-templates />
   </body>
  </html>
 </xsl:template>

 <xsl:template match="libro">
  <img>
   <xsl:attribute name="src">
    <xsl:value-of select="fotografia" />
   </xsl:attribute>
  </img>
 </xsl:template>

</xsl:stylesheet>
```

Resultado

```
<?xml version="1.0" encoding="UTF-
8"?>
<html>
        <body>
                <img
                src="cervantes.png"/>
        </body>
</html>
```

10. XML para probar las estructuras de control

A partir de ahora, utilizaremos el siguiente documento XML con el que probaremos los códigos de las distintas transformaciones:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="prueba.xsl" ?>

<notas>
        <alumno convocatoria="Septiembre">
                <nombre>Carlos</nombre>
                <apellidos>Amaya Arozamena</apellidos>
                <matricula>m019843</matricula>
                <cuestionarios>8.0</cuestionarios>
                <tareas>8.0</tareas>
                <examen>6.0</examen>
                <final>8.0</final>
         /alumno>…

Como es de suponer, alumno se repite varias veces.
```

Además, para simplificar los códigos, en los siguientes apartados sólo se mostrarán los códigos que deben aparecer entre las etiquetas <body> … </body> del siguiente código:

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

<xsl:template match="/">
        <html>
                <head><title>Práctica de transformación</title></head>

                <body>
                    X
                </body>

        </html>
</xsl:template>
</xsl:stylesheet>
```

11. Iteración: <xsl:for-each>

La instrucción **<xsl:for-each>** (Sintaxis: **<xsl:for-each select="expresión">**) permite procesar de forma iterativa (repetitiva) diversos nodos del documento XML origen. Estos nodos se referencian mediante una expresión XPath.

XSLT	Transformación
<xsl:for-each select="/notas/alumno"> <xsl:value-of select="**nombre**"/> <xsl:text> </xsl:text> <xsl:value-of select="**apellidos**"/>	<body> Carlos Amaya Arozamena José Muñoz Soto Ana Martínez de la Fuente Roberto Carrera Fernández

 </xsl:for-each>	Concepción Lalinde Priego Esther Pereda </body>

Como se puede observar, esta instrucción necesita el uso de una expresión XPath para indicar sobre qué elemento debe realizarse esa iteración o repetición de instrucciones.

Vamos a trabajar el código anterior de dos maneras.

- Modifica la transformación anterior añadiendo un nuevo elemento, **notas**. Pero ahora, haz que todo aparezca dentro de una tabla.
- Y por último, **para que empieces a familiarizarte con las expresiones XPath**, crea una nueva columna en la tabla para que muestre el atributo **convocatoria**.

La expresión XPath que te posiciona sobre el atributo (teniendo en cuenta que ya estás en el elemento alumno) es: **@convocatoria**. Es decir:

<xsl:value-of select="@convocatoria"/>

12. Condiciones simples: <xsl:if >

La instrucción **<xsl:if>** (Sintaxis: **<xsl:if select="condición">**) permite determinar si una acción se llevará a cabo o no.

La condición es la que determina si los elementos hijos de éste se tendrán en cuenta o no.

Por ejemplo, en el siguiente ejemplo sólo se muestra el **nombre** y **apellidos** de aquellos alumnos que tienen en el atributo **convocatoria** el valor de **Junio**. Utilizaremos esta orden combinándola con la instrucción **<xsl:for-each>**

XSLT	Transformación
<xsl:for-each select="/notas/alumno"> **<xsl:if** **test="@convocatoria='Junio'">** <xsl:value-of select="nombre"/> <xsl:text> </xsl:text> <xsl:value-of select="apellidos"/> **</xsl:if>** </xsl:for-each>	<body> José Muñoz Soto Ana Martínez de la Fuente Esther Pereda </body>

Llegados a este punto vamos a utilizar los siguientes operadores relacionales con los que debes ir familiarizándote en el uso de las condiciones XSLT. Y alguna cosilla más sobre expresiones XPath.

- = (igual)
- != (no igual)
- < (menor que)
- > (mayor que)

Prueba el siguiente código y comprueba que hace exactamente lo mismo que el anterior.

```
<xsl:for-each select="/notas/alumno[@convocatoria='Junio']">
        <p>
                <xsl:value-of select="nombre"/>
                <xsl:text> </xsl:text>
                <xsl:value-of select="apellidos"/>
        </p>
</xsl:for-each>
```

Utilizando el primer código, cambia la condición para obtener el nombre, apellidos y nota del alumno con nota **igual a 8.0** (<xsl:if test="final=8.0">). Después, haz modificaciones para obtener los datos de los que tienen notas **distintas a 8.0**, **menor que 8.0** y,por último, **mayores que 5.0**.

13. Condiciones compuestas: <xsl:choose >, <xsl:when> y <xsl:otherwise>

Pero qué ocurre si cuando no se cumple una condición se quiere expresar otra opción. Ésta es la principal diferencia entre las condiciones simpes y las compuestas. Su sintaxis es la siguiente:

```
<xsl:choose>
        <xsl:when test="condición">
            ✘
        </xsl:when>
        <xsl:when test="condición">
            ✘
        </xsl:when>
        <xsl:when test="condición">
            ✘
        </xsl:when>
        <xsl:otherwise>
            ✘
        </xsl:otherwise>
</xsl:choose>
```

Podemos tener tantas condiciones como queramos. De forma secuencial una tras otra, se irán comprobando sus valores. Si ninguna de ellas se cumple, se tendrá en cuenta la existencia de un elemento **<xsl:otherwise>**. Si existiese, ésa sería la acción a llevar a cabo. En caso contrario, no se llevaría a cabo ninguna acción. **Observa el siguiente código, la transformación y el comentario final sobre operadores lógicos.**

XSLT

```
<xsl:for-each select="/notas/alumno">
    <p>
        <xsl:value-of select="nombre"/><xsl:text>-</xsl:text>
        <xsl:value-of select="apellidos"/><xsl:text>-</xsl:text>
        <xsl:value-of select="final"/><xsl:text>-</xsl:text>
        <xsl:choose>
            <xsl:when test="final&gt;='9.0'">Sobresaliente</xsl:when>
            <xsl:when test="final&gt;='7' and final&lt;'9'">Notable</xsl:when>
            <xsl:when test="final&gt;='6' and final&lt;'7'">Bien</xsl:when>
            <xsl:when test="final&gt;='5' and final&lt;'6'">Suficiente</xsl:when>
            <xsl:otherwise> SUSPENSO</xsl:otherwise>
        </xsl:choose>
    </p>
</xsl:for-each>
```

Transformación

```
<body>
        <p>Carlos-Amaya Arozamena-8.0-Notable</p>
        <p>José-Muñoz Soto-8.5-Notable</p>
        <p>Ana-Martínez de la Fuente-5.5-Suficiente</p>
        <p>Roberto-Carrera Fernández-6.5-Bien</p>
        <p>Concepción-Lalinde Priego-3.0- SUSPENSO</p>
        <p>Esther-Pereda-2.5- SUSPENSO</p>
```

```
</body>
```

Al igual que antes comentamos los operadores relacionales que puedes utilizar. Éstos son los operadores lógicos que puedes utilizar en las expresiones condicionales y que hemos utilizado en el código anterior: **and**, **or** y **not()**.

14. Ordenación: <xsl:sort >

La instrucción **<xsl:sort>** (Sintaxis: **<xsl:sort select="expresión">**) es la que se utiliza para cuando queremos que la información aparezca ordenada según el valor de algún elemento.

Vamos a aprovechar el código anterior para mostrar las líneas ordenadas alfabéticamente según el valor de los apellidos. **Es suficiente con añadir una instrucción justo después del bucle for-each**. En concreto, **<xsl:sort select="apellidos"/>**.

XSLT

```
<xsl:for-each select="/notas/alumno">
    <xsl:sort select="apellidos"/>
    <p>
        <xsl:value-of select="apellidos"/><xsl:text>, </xsl:text>
        <xsl:value-of select="nombre"/><xsl:text>-</xsl:text>
        <xsl:value-of select="final"/><xsl:text>-</xsl:text>
        <xsl:choose>
                <xsl:when test="final&gt;='9.0'">Sobresaliente</xsl:when>
                <xsl:when test="final&gt;='7' and final&lt;'9'">Notable</xsl:when>
                <xsl:when test="final&gt;='6' and final&lt;'7'">Bien</xsl:when>
                <xsl:when test="final&gt;='5' and final&lt;'6'">Suficiente</xsl:when>
                <xsl:otherwise> SUSPENSO</xsl:otherwise>
        </xsl:choose>
    </p>
</xsl:for-each>
```

Transformación

```
<body>
        <p>Amaya Arozamena, Carlos-8.0-Notable</p>
        <p>Carrera Fernández, Roberto-6.5-Bien</p>
        <p>Lalinde Priego, Concepción-3.0- SUSPENSO</p>
        <p>Martínez de la Fuente, Ana-5.5-Suficiente</p>
        <p>Muñoz Soto, José-8.5-Notable</p>
        <p>Pereda, Esther-2.5- SUSPENSO</p>
</body>
```

15. Conclusión

Durante esta unidad hemos aprendido que es posible cambiar la apariencia de nuestros documentos XML. Podemos mostrar todos los datos del documento, o sólo algunos, cambiar su apariencia, etc.

Siendo mucho lo que hemos probado, aún existen más cosas que pueden hacerse con XSL. Ahora que ya conoces lo básico de este sistema de transformaciones está preparado para aforntar sin miedo cualquier otro aspecto de las mismas.

Para más información sobre este tema, no dudes en acudir a la página del W3C dedicada a las transformaciones.

- Noviembre de 1999: XSLT 1.0 (https://www.w3.org/TR/1999/REC-xslt-19991116)
- Marzo de 2021: XSLT 2.0 (https://www.w3.org/TR/xslt20/)
- Junio de 2017: XSLT 3.0 (https://www.w3.org/TR/xslt-30/)