5 Conversión y adaptación de documentos XML

OBJETIVOS DEL CAPÍTULO

- ✓ Conocer XSL como herramienta de transformación de documentos XML.
- ✓ Saber qué elementos básicos pueden utilizarse en la definición de estilos XSL.
- ✓ Utilización de plantillas para facilitar el proceso de conversión y transformación de documentos XML.
- ✓ Conocer qué operadores permiten discriminar los datos para aplicar estilos diferentes.
- ✓ Realizar ejercicios prácticos que afiancen los conceptos de la transformación a través de XSL.

Tal y como se ha explicado en capítulos anteriores, 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.

Aunque podemos visualizar un fichero XML con un simple editor de texto, no es la manera más amigable ni profesional para presentar los datos que están almacenados en su interior. Es ahí donde debemos utilizar alguna herramienta que nos permita convertir y transformar los datos en el formato que deseemos. Esa herramienta se llama XSL (*Extensible Stylesheet Language*).

XSL es a XML, lo que las hojas de estilo en cascada (CSS) a HTML. XSL permite tomar pleno control sobre los datos, pudiendo establecerse criterios como qué datos ver, en qué orden visualizarlos, estableciendo filtros y definiendo formatos de salida para su representación. Es, por tanto, una herramienta de procesado muy potente que hay que conocer en profundidad.

Para entrar dentro del contexto de la conversión y transformación de ficheros XML, se define el siguiente ejemplo con código XML que permitirá almacenar un conjunto de libros en una librería:

EJEMPLO 5.1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<libreria>
    <libro>
    <titulo>Physics-based animation</titulo>
    <autor>Kenny Erleben</autor>
    <editor>Charles River Media</editor>
    <isbn>978-1584503804</isbn>
    <precio>50.06</precio>
    </libro>
    <libro>
    <titulo>Principios de seguridad informática para usuarios</titulo>
    <autor>Carlos Garre</autor>
    <editor>Dykinson</editor>
    <isbn>978-84-9849-998-8</isbn>
    <precio>13.00</precio>
    </libro>
    <libro>
    <titulo>Ejercicios complementarios de lógica digital</titulo>
    <autor>Alberto Sánchez</autor>
    <editor>Dykinson</editor>
    <isbn>978-84-9849-703-8</isbn>
    <precio>12.00</precio>
    </libro>
</libreria>
```

Puede verse claramente que la librería puede contener un conjunto de libros. Cada libro tiene una serie de atributos propios como puede ser el título, autor o ISBN. En el ejemplo anterior se han almacenado tres libros y se quiere visualizarlos de una manera más amigable. Se podría definir una XSL como la siguiente:

EJEMPLO 5.2

```
<?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>
      <h1>Mi biblioteca</h1>
         <table>
         <tr bgcolor="#887788">
            <th>Título</th>
            <th>Autor</th>
         </tr>
         <xsl:for-each select="libreria/libro">
         <tr>
            <td><xsl:value-of select="titulo"/></td>
            <td><xsl:value-of select="autor"/></td>
         </tr>
         </xsl:for-each>
      </table>
      </body>
      </html>
</xsl:template>
</xsl:stylesheet>
```

Esta hoja de estilo XSL define que por cada libro almacenado en la librería, se añade una entrada en una tabla HTML con los valores del título y el nombre del autor. El resultado de vincular al XML una hoja de estilo XSL permite obtener un resultado como el siguiente:

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre
Ejercicios complementarios de lógica digital	Alberto Sánchez

Figura 5.1. *Aplicación de un XSL básico*

Este es un pequeño ejemplo de conversión. En principio se puede transformar un documento XML a otro XML o cualquier otro distinto que pueda ser reconocido por un navegador web (HTML o XHML).

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

1. Tener bien definido el documento XML.
2. Crear una hoja de estilo XSL bien formada.
3. Vincular al documento XML la hoja de estilo XSL.

Los dos primeros puntos asumimos que se cumplen al estar determinados por los DTDs o esquemas definidos anteriormente. Además se indica sobre qué espacio de nombres se trabaja por lo que es fácil cumplir las dos condiciones iniciales. Para vincular en un documento XML una hoja de estilo XSL cualquiera, por ejemplo "tablaBiblioteca.xsl", es tan simple como añadir al fichero XML lo siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="tablaBiblioteca.xsl"?>
 <libreria>
 ...
 </libreria>
```

Cumplidos los pasos anteriores, el procesador de documentos XSL comienza a recorrer el árbol del documento XML. Se inicia por el nodo raíz y se irá recorriendo todo el árbol haciendo que cada parte del documento XML pueda tener un formato específico (árbol de resultados). Pero, ¿cómo determinar qué formatos tendrán cada una de las partes del documento?

La respuesta a la pregunta es la definición de plantillas en la hoja de estilos. Una plantilla es un patrón que cuando se cumple puede generar resultados de formateo en el árbol final. Es decir, con un documento de origen que se recorre por completo buscando concordancias con las plantillas definidas, de manera que si se cumple alguna, se aplica el formato definido y se genera en el árbol del documento final.

ACTIVIDADES 5.1

- Insertar varios libros más en el fichero XML indicado cuyo coste sea inferior a 10.50 €.
- Insertar el dato número de páginas en cada libro contenido en el fichero XML inicial. Por ejemplo, el elemento será <numPaginas>150<numPaginas>.
- Cambiar el título de la tabla a "Mi biblioteca personal".
- Cambiar el color de la tabla en el fichero XSL.
- Añadir una columna más al final en la que se muestre el precio del libro.
- Añadir una columna más al principio en la que se muestre el ISBN.

5.1 TRANSFORMACIÓN DE DOCUMENTOS (XSL)

XSL es un lenguaje que nos permite definir un conjunto de reglas que, aplicados sobre un documento XML, permite transformarlo en un resultado formateado más adecuado a nuestros intereses. Para ello, se necesita almacenar esas reglas dentro de un fichero llamado hoja de estilo XSL.

El fichero que almacena la hoja de estilo XSL, como fichero XML que es, debe estar bien formado. Usando el ejemplo anterior de los libros, se puede ver que el fichero XSL posee una declaración inicial que determina su contenido como XML y su codificación:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

A continuación se indica la etiqueta que identifica al fichero XSL y el espacio de nombres en el que se basa. En este caso tenemos dos opciones:

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

o

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

Las dos maneras están soportadas en el estándar establecido en el consorcio W3C y se han elevado a recomendaciones de uso. Se puede observar que, independientemente de la que se utilice, ambas hacen referencia a un espacio de nombres común en su versión 1.0.

La siguiente etiqueta interesante es la que nos permite definir un elemento plantilla dentro del XSL. Todo lo que quede entre las siguientes etiquetas:

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

será lo que permitirá generar la salida formateada. Cabe destacar que con esta etiqueta se puede indicar sobre qué parte del documento XML se quiere actuar, utilizando el atributo "*match*" para ello. En este caso se ha querido actuar sobre la raíz del propio documento XML.

5.2 ELEMENTOS BÁSICOS

5.2.1 XSL:FOR-EACH

En este momento tenemos en el fichero XML un conjunto de libros bien identificados. Es posible que al usuario final le interese recorrerse todos y cada uno de los libros y extraer los datos para darles un nuevo formato. En este caso, la siguiente etiqueta indica que se recorran todo el conjunto de elementos XML que sean libros:

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

5.2.2 XSL:VALUE-OF

Finalmente queda extraer el contenido del libro. Para ello la etiqueta que extrae la información de ese elemento XML seleccionado es la siguiente:

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

Esta es la manera más sencilla de realizar un listado completo, formateado y con estilos adecuados, de todos los libros almacenados en la biblioteca. Aunque este ejemplo es muy simple, la potencia de las hojas de estilo XSL no acaba con estos elementos.

5.2.3 XSL:SORT

En este momento el lector podría indicar que tiene tantos libros almacenados que resultaría muy interesante mostrarlos de manera ordenada (y no tal y como se salvaron en el fichero XML). Si lo que interesa es ordenarlos por el título del libro, tras la etiqueta "*for-each*", incluimos ésta:

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

Mi biblioteca

Título	Autor
Ejercicios complementarios de lógica digital	Alberto Sánchez
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre

Figura 5.2. *Aplicar ordenación por título*

Si por el contrario, el orden adecuado es por autor, entonces se debería incluir la siguiente:

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

Mi biblioteca

Título	Autor
Ejercicios complementarios de lógica digital	Alberto Sánchez
Principios de seguridad informática para usuarios	Carlos Garre
Physics-based animation	Kenny Erleben

Figura 5.3. *Aplicar ordenación por autor*

Lo realmente interesante es que se puedan realizar búsquedas con determinados patrones preestablecidos. En este caso, parece lógico pensar que en vez de recorrerse todos los elementos, deberíamos filtrar por ese patrón (independientemente del orden que establezcamos). Si, por ejemplo, queremos extraer todos los libros del autor "*Kenny Erleben*", se podría cambiar lo siguiente:

```
<xsl:for-each select="libreria/libro[autor='Kenny Erleben']">
...
</xsl:for-each>
```

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben

Figura 5.4. *Búsqueda por autor "Kenny Erleben"*

Si, por el contrario, lo que interesa es realizar una búsqueda con otro operador, como por ejemplo, todos los libros en los que el autor no es "*Carlos Garre*", ordenado por autor, el cambio sería tan simple como éste:

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

Mi biblioteca

Título	Autor
Ejercicios complementarios de lógica digital	Alberto Sánchez
Physics-based animation	Kenny Erleben

Figura 5.5. *Búsqueda por autor distinto a "Carlos Garre"*

ACTIVIDADES 5.2

- Sin ningún patrón de búsqueda establecido, ordenar los resultados por precio.
- Insertar en el fichero XML que se indica inicialmente, un nuevo libro cuyo autor sea el mismo a uno ya incluido anteriormente.
- Buscar los libros del autor anterior y comprobar que salen todos sus libros (ordenados por precio).
- Buscar todos los libros que no son del autor anterior. Comprobar que en los resultados faltan sus libros.

5.3 OPERADORES EN XSL

5.3.1 ELEMENTO XSL:IF

Hasta este momento, se ha visto operadores de igualdad y desigualdad para cambiar el patrón de búsqueda pero no son los únicos que existen. Podrían usarse también los siguientes:

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 (>): ">"

En ocasiones se necesita mayor potencia a la hora de establecer un patrón de búsqueda. Los operadores lógicos vistos anteriormente son efectivos para filtros simples pero operadores lógicos no es suficiente. Existe la posibilidad de indicar, con el elemento "*<xsl:if>*", condiciones más complejas en la evaluación del fichero XML. La sintaxis de este nuevo elemento es la siguiente:

```
<xsl:if test="expresion">
...
</xsl:if>
```

Imaginemos que queremos mostrar todos aquellos libros cuyo coste sea superior a 12 euros. Para ello, solo modificaremos las siguientes líneas del fichero XSL:

```
<xsl:for-each select="libreria/libro">
<xsl:if test="precio &gt; 12">
...
</xsl:if>
</xsl:for-each>
```

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre

Figura 5.6. *Búsqueda por coste de libro mayor que 12 €*

ACTIVIDADES 5.3

- Añadir al fichero XML inicial (ejemplo 5.1), dos libros de un nuevo autor. Sus precios serán 10.50 € y 12.50 € respectivamente
- Realizar un fichero XSL que busque todos aquellos libros cuyo coste sea superior a 10 €. Asegurarse que aparece los libros del autor nuevo.
- En el mismo fichero XSL anterior, añadir una condición en la que solo se muestren aquellos cuyo autor sea el último que se ha definido.
- Cambiar la condición del coste del libro. Ahora solo se quiere aquellos libros cuyo coste sea superior a 12 €. Comprobar que solo aparece un libro de los dos anteriores añadidos.

5.3.2 ELEMENTO XSL:CHOOSE

El elemento "*<xsl:if>*" no es la única manera de condicionar los resultados. Existe otro elemento que nos permite establecer múltiples condiciones dentro del recorrido en el árbol XML. El elemento "*<xsl:choose>*" es muy similar a la instrucción "*switch*" de C/C++ o "*case*" de Pascal. Se pueden establecer tantas expresiones condicionales como se quieran mediante los elementos "*<xsl:when>*". Si se quiere establecer una condición por defecto, el elemento a utilizar sería "*<xsl:otherwise>*". La sintaxis quedaría como sigue:

```
<xsl:choose>
 <xsl:when test="expresion">
 ...
 </xsl:when>
 <xsl:when test="expresion">
 ...
 </xsl:when>
 <xsl:otherwise>
 ...
 </xsl:otherwise>
</xsl:choose>
```

Siguiendo con el ejemplo inicial, establezcamos una condición nueva en la que si el libro es menor de 12.50 euros muestre en color rojo todos sus datos. Si es mayor de 25.50 euros, en color verde. Y si no se cumple ninguna de las anteriores, en color azul. Esto se realizaría de la siguiente manera:

```
<xsl:for-each select="libreria/libro">
 <tr>
  <xsl:choose>
   <xsl:when test="precio &lt; 12.50">
    <td bgcolor="#ff0000">
     <xsl:value-of select="titulo"/>
    </td>
    <td bgcolor="#ff0000">
     <xsl:value-of select="autor"/>
    </td>
   </xsl:when>
   <xsl:when test="precio &gt; 25.50">
    <td bgcolor="#00ff00">
     <xsl:value-of select="titulo"/>
    </td>
    <td bgcolor="#00ff00">
     <xsl:value-of select="autor"/>
    </td>
   </xsl:when>
   <xsl:otherwise>
    <td bgcolor="#0000ff">
     <xsl:value-of select="titulo"/>
    </td>
    <td bgcolor="#0000ff">
     <xsl:value-of select="autor"/>
    </td>
   </xsl:otherwise>
  </xsl:choose>
 </tr>
</xsl:for-each>
```

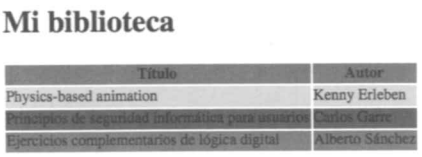

Figura 5.7. Cambio de estilo condicional con xsl:choose

ACTIVIDADES 5.4

- Añadir al fichero XML inicial (ejemplo 5.1), dos libros de un nuevo autor. Sus precios serán 10.50 € y 60.50 € respectivamente
- Comprobar con el fichero XSL indicado que los nuevos libros se pintan del color indicado.
- Añadir las columnas precio al final e ISBN al principio de la tabla de resultados.
- Cambiar los colores para que si el coste es mayor que 25 €, las filas sean de color rojo y verde si es menor que 25 €.

5.4 LAS PLANTILLAS

5.4.1 ELEMENTO XSL:TEMPLATE

Hasta el momento solo hemos utilizado una única plantilla para todo el documento XML. Se habló del elemento "*<xsl:template>*" como etiqueta que permitía definir sobre qué parte del documento XML se quería actuar. Si se observan con detenimiento todos los ejemplos anteriores, el atributo "*match*" referenciaba siempre la raíz del documento XML, no estableciendo distinción entre otras ramas o elementos que lo componen. A continuación se muestra un cambio radical en el fichero XSL para que, usando distintas plantillas, podamos ver el nombre del autor en un guis claro y el título del libro en otro más oscuro (sin ningún elemento condicional por simplicidad del ejemplo):

```
<?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> <body>
    <h1>Ejemplo Plantillas</h1>
    <xsl:apply-templates/>
  </body> </html>
</xsl:template>
```

```
<xsl:template match="libreria">
 <h2>Mi biblioteca</h2>
 <table>
  <tr bgcolor="#887788">
   <th>Título</th> <th>Autor</th>
  </tr>
  <xsl:apply-templates select="libro"/>
 </table>
</xsl:template>

<xsl:template match="libro">
 <tr>
  <td><xsl:apply-templates select="titulo"/></td>
  <td><xsl:apply-templates select="autor"/></td>
 </tr>
</xsl:template>

<xsl:template match="titulo">
 <td bgcolor="#DDEEDD"><xsl:value-of select="."/></td>
</xsl:template>

<xsl:template match="autor">
 <td bgcolor="#AABBAA"><xsl:value-of select="."/></td>
</xsl:template>

</xsl:stylesheet>
```

Ejemplo Plantillas

Mi biblioteca

Título	Autor
Physics-based animation	Kenny Erleben
Principios de seguridad informática para usuarios	Carlos Garre
Ejercicios complementarios de lógica digital	Alberto Sánchez

Figura 5.8. *Aplicación de plantilla*

En este ejemplo se puede observar que se han establecido cinco plantillas personalizadas para los elementos "/", "*libreria*", "*libro*", "*titulo*" y "*autor*". Cuando se llame a esta hoja de estilos XSL desde un documento XML, se intentará reconocer los patrones de las plantillas, sustituyendo en la salida los contenidos que se indiquen por cada plantilla.

Se debe destacar una serie de elementos que no se han explicado anteriormente y que permiten la aplicación directa de la plantilla en el documento final:

- <*xsl:apply-templates*/>: Indica que se apliquen el resto de las plantillas definidas en cuanto se cumplan el patrón "match" de cada una.
- <*xsl:apply-templates select="XXXX"*/>: Indica que se aplique en ese momento, el contenido de la plantilla "*XXXX*" definida en el documento XSL.

- *<xsl:value-of select "."/>*: En el nodo XML actual de la plantilla actual, indica que se copie el valor que contenga ese nodo. Por ejemplo, si el nodo actual es de tipo "*titulo*", insertará como resultado el título del libro actual.

El proceso seguido cuando se recorre un documento XML desde una transformación XSL, siempre es el mismo. Intuitivamente comienza a recorrerse el fichero XML desde su nodo raíz, aplicando en ese momento la plantilla asociada. En el ejemplo anterior se utilizará plantilla "/", generando un esqueleto de documento HTML válido. El elemento "*<xsl:apply-templates/>*" que está situado en la plantilla "/", indica que se recorra todos y cada uno de los nodos definidos en el fichero XML aplicando las plantillas definidas. Si en ese recorrido del documento XML se encuentra el elemento "*libreria*", aplicará la plantilla del mismo nombre definida en el fichero XSL.

Es importante reseñar que cuando se aplica la plantilla librería, se establecen las propiedades y etiquetas HTML necesarias para crear una tabla de datos. Esta tabla contendrá datos en su interior por lo que el elemento "*<xsl:apply-templates select="libro"/>*" indicará que, cuando se reconozca un nodo XML "libro", se aplique la plantilla definida para ello. La misma explicación tienen la aplicación de las plantillas "*titulo*" y "*autor*".

Finalmente, si en el recorrido del documento XML se encuentra sobre un nodo "*titulo*" o "*autor*", se aplicará su plantilla correspondiente, con la salvedad que en esos nodos se requiere la extracción de la información para el documento final. Esa funcionalidad vendrá dada por el elemento "*<xsl:value-of select="."/>*", copiando el valor de ese nodo y enviándolo al documento HTML final.

Podemos ver, por tanto, el recorrido del documento XML como un conjunto de llamadas a "*procedimientos*" (plantillas), que según el caso, integrarán datos de salida en el documento final.

ACTIVIDADES 5.5

- Insertar el dato número de páginas en cada libro contenido en el fichero XML inicial. Por ejemplo, el elemento será <numPaginas>150<numPaginas>.
- Añadir las columnas precio y número de páginas al final de la tabla de resultados.
- Añadir la columna ISBN al principio de la tabla de resultados.
- Añadir al fichero XSL una plantilla nueva para cambiar el estilo del elemento "numPaginas" para que se muestre en rojo si el libro tiene más de 150 páginas.
- Comprobar los resultados obtenidos comparándolo con los datos almacenados en el fichero XML.

5.5 CASO PRÁCTICO

Se propone crear un fichero XML nuevo que almacene datos de un videoclub. El videoclub almacenará los siguientes datos de una película:

- ✓ Título.
- ✓ Nacionalidad.
- ✓ Productor.
- ✓ Director.
- ✓ Año.
- ✓ Duración.
- ✓ Género.
- ✓ Sinopsis.
- ✓ Foto.
- ✓ URL.

Se deben crear al menos 20 películas distintas en las que algunas sean de la misma nacionalidad y del mismo director.

El valor del elemento foto será el nombre de un fichero .jpg o .png que deberá estar descargado en el mismo sitio donde está el fichero XML.

Se pide:

- Generar un fichero XSL ("p1.xsl") en el que se muestre una tabla con todos los datos de las películas (salvo la foto) que el videoclub tiene almacenado.
- Generar un fichero XSL ("p2.xsl") que muestre titulo, director y la foto (no el nombre del fichero sino la imagen en sí) de todas las películas almacenadas en el videoclub.
- Generar un fichero XSL ("p3.xsl") igual que el anterior pero realizado con plantillas y que al hacer clic en la foto, nos lleve a la URL almacenada.

RESUMEN DEL CAPÍTULO

En este capítulo se han explicado los conceptos básicos para la transformación de documentos XML mediante el lenguaje de transformación XSL. XSL es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio. Esto ha permitido saber qué elementos básicos son los que componen cualquier documento XSL (plantillas, elementos para recorrer los nodos, elementos para extraer la información, para ordenar los resultados, de tipo condicional, operadores, etc.).

También se ha proporcionado un conjunto de ejemplos y actividades pequeñas que permiten comprobar la funcionalidad de las transformaciones de manera fácil y sencilla. Se ha propuesto un caso práctico, de mayor complejidad, en el que se deben poner en práctica la práctica totalidad de las transformaciones mostradas anteriormente, fijando los conceptos como la generación de distintos documentos XSL para mostrar distintos resultados y la aplicación de las plantillas.

EJERCICIOS PROPUESTOS

1. Busque por Internet cuál es la última recomendación del W3C de las transformaciones XSL.

2. Busque por Internet cuándo utilizar CSS y cuándo utilizar XSL (pista: la W3C responde esta pregunta).

3. Crear un documento XML que almacene una lista de CDs de música. Cada CD deberá almacenar la siguiente información:

- Título del album.
- Artista.
- Títulos de las canciones con el tiempo por canción.
- Sello discográfico.
- Año de publicación.

Al menos insertar 10 CDs.

4. Con el documento XML anterior, se pide:

- Generar un fichero XSL ("cd_p1.xsl") en el que se muestre una tabla con todos los datos de los discos de música.
- Elegir un artista cualquiera y generar un fichero XSL ("cd_p2.xsl") en el que se muestre una tabla con todas las canciones de ese artista.
- Elegir un sello cualquiera y generar un fichero XSL ("cd_p3.xsl") en el que se muestre una tabla con todas las canciones de ese sello discográfico.
- Elegir una duración máxima de canción y generar un fichero XSL ("cd_p4.xsl"), en el que se muestre una tabla con todas las canciones que tienen una duración inferior a la elegida.