MEDAC
Instituto Oficial de Formación Profesional

LENGUAJE DE MARCAS Y SISTEMA DE GESTIÓN DE INFORMACIÓN

Transformaciones XSLT (II)

14

ÍNDICE

OBJETIVOS

Practicar con las transformaciones XSLT.

Identificar los principales elementos XSLT.

Conocer herramientas de procesamiento y depuración.

Conocer XSL-FO.

/ 1. Introducción y contextualización práctica

Seguimos trabajando con instrucciones XSLT, conociendo más elementos y estudiando en profundidad cómo se pueden recorrer los nodos de un fichero XML y tomar decisiones en función del contenido de esos nodos.

Los resultados los vamos a poder ordenar por criterios definidos, y vamos a utilizar herramientas de procesamiento y depuración para poder identificar posibles errores.

El editor que se ha usado en los ejemplos es Oxygen XML Editor, que dispone de una versión gratuita de 30 días.

El instituto de FP Málaga Formación sigue confiando en Webmalagadesign y ahora necesitan formatear en forma de tabla las notas de los alumnos, destacando aprobados y suspensos.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y resolución del caso práctico.

Fig. 1. Logo de Oxygen XML editor.

Audio intro. "Notas de alumnos"
https://bit.ly/3jmgO1k

/ 2. Elementos XSLT: <xsl:element> y <xsl:attribute>

En XSLT, se pueden añadir elementos y atributos nuevos respecto al XML de origen. Los valores que se le pueden dar a estos nuevos elementos o atributos pueden ser constantes o depender de otros elementos del fichero XML.

```
<xsl:element name="nombre">
    <!-- Contenido de la plantilla -->
</xsl:element>
```

Código 1. Sintaxis <xsl:element>.

Atributos:

- **name:** nombre del nuevo nodo.

Si queremos añadir atributos, la sintaxis sería:

```
<xsl:attribute name="nombre">
    <!-- Contenido de la plantilla -->
</xsl:attribute>
```

Código 2. Sintaxis <xsl:attribute>.

Atributos:

- **name:** Nombre del nuevo atributo.

Para aplicar estos conceptos, se va a añadir un nuevo elemento a la carta de desayunos, en concreto, el precio con IVA, y también se va a convertir el elemento calorías en un atributo del nodo desayuno. Veámoslo en los siguientes ejemplos. (En XSLT también se pueden utilizar variables. Aquí podemos encontrar información sobre el uso de variables).

```
<?xml version="1.0"  encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:output method="xml" indent="yes"/>
    <xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="desayuno">
        <desayuno>
            <xsl:attribute name="calorias">
                <xsl:value-of select="calorias"/>
            </xsl:attribute>
            <nombre>
                <xsl:value-of select="nombre"/>
            </nombre>
            <precio><xsl:value-of select="precio"/>
            </precio>
            <descripcion><xsl:value-of select="descripcion"/>
            </descripcion>
            <xsl:element name="precio_con_iva">
                <xsl:value-of select="precio*1.16"/>
            </xsl:element>
        </desayuno>
    </xsl:template>
</xsl:stylesheet>
```

Código 3. Fichero XSLT.

```
<?xml version="1.0" encoding="UTF-8"?>
<desayuno calorias="250">
   <nombre>Smoothie de fruta</nombre>
   <precio>4.00</precio>
   <descripcion>Fresa, plátano y mango.</descripcion>
   <precio_con_iva>4.64</precio_con_iva>
</desayuno>
```

Código 4. Resultado obtenido para el primer desayuno

/ 3. Elemento XSLT: <xsl:for-each>

Esta instrucción se utiliza para recorrer de manera iterativa diversos nodos del documento origen y aplicarles la transformación necesaria. La sintaxis es:

```
<xsl:for-each select="expression">
    <!-- Contenido de la plantilla -->
</xsl:for-each>
```

Código 5. Sintaxis <xsl:for-each>.

Atributos:

- **select:** Expresión XPath que indica los nodos que se van a recorrer.

En el ejemplo del código 7, se construye una página web y se utiliza la instrucción **<xsl:for-each>** para recorrer todos los nodos desayuno. Dentro de esta instrucción, se van mostrando los nombres y precios del desayuno. Estos valores se van incluyendo en una tabla.

Como se observa, se han incluido estilos en la propia página HTML. Hay que tener en cuenta que también es posible incluir el estilo en un fichero independiente, tal y como se explicó en el tema de CSS. Simplemente, habría que incluir la línea siguiente en el <head>:

```
<link rel="stylesheet" type="text/css" href="urlhojaestilos.css">
```

Código 6. Incluir hoja de estilos externa.

```
<?xml version="1.0"  encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/carta">
        <html>
            <head>
                <style>
                    table{border:solid 2px black;
                    box-shadow:10px 10px 10px grey;}
                    th{color:white;background-color:#159dcc;}
                    td{border:solid 2px black;}
                </style>
            </head>
            <body>
                <table>
                    <th>Nombre</th>
                    <th>Precio</th>
                    <xsl:for-each select="desayuno">
                        <tr>
                            <td><xsl:value-of select="nombre"/></td>
                            <td><xsl:value-of select="precio"/></td>
                        </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Código 7. Uso de for-each

Nombre	Precio
Smoothie de fruta	4.00
Gobres Belgas con fresas	7.95
Huevos revueltos con Tostadas	6.95
Tostada Francesa	4.50
Tostón con Salmón	6.95

Fig. 2. Resultado obtenido

/ 4. Elementos XSLT: <xsl:if> y <xsl:choose>

- **<xsl:if>:** Se trata de una instrucción condicional que añadirá los resultados en el archivo si se cumple una condición. Su sintaxis es:

```
<xsl:if test="expresion">
    ...salida cuando la expresion es cierta...
</xsl:if>
```

Código 8. Sintaxis de <xsl:if>.

Atributos:

- **test:** Se indica la condición que se tiene que comprobar. Si se utilizan los caracteres '>' o '<' para las comprobaciones, se deben de usar las entidades '>' y '<', respectivamente.

  Por ejemplo, si queremos incluir solo los desayunos que tiene un precio inferior a 5€, este sería el código XSLT:

```
<table>
    <th>Nombre</th>
    <th>Precio</th>
    <xsl:for-each select="desayuno">
        <xsl:if test="precio &lt; 5">
            <tr>
                <td><xsl:value-of select="nombre"/></td>
                <td><xsl:value-of select="precio"/></td>
            </tr>
        </xsl:if>
    </xsl:for-each>
</table>
```

Código 9. Uso de <xsl:if>.

Nombre	Precio
Smoothie de fruta	4.00
Tostada Francesa	4.50

Fig. 3. Resultado obtenido

- **<xsl:choose>:** Esta instrucción permite elegir entre varias condiciones para realizar diferentes transformaciones. Se usa en conjunción con los elementos <xsl:otherwise> y <xsl:when>. La sintaxis es:

```
<xsl:choose>
    <xsl:when test="expression">...opción 1...</xsl:when>
    <xsl:when test="expression">...opción 2...</xsl:when>
    <xsl:otherwise>...en otro caso...</xsl:otherwise>
</xsl:choose>
```

Código 10. Sintaxis de <xsl:choose>.

El funcionamiento se basa en ir comprobando cada una de las condiciones, y aquella que se cumpla, ejecutará la transformación indicada. En el caso de que no se cumpla ninguna condición, se ejecuta el contenido del elemento <xsl:otherwise>.

Siguiendo con el ejemplo de los desayunos, en el código adjunto, se codifica cómo mostrar los desayunos que tengan un precio inferior a 5€ con un fondo gris; los que tengan un precio superior a 7, de color amarillo, y el resto, de color blanco. El resultado se muestra en la siguiente figura y el código completo en el **anexo** del tema.

Nombre	Precio
Smoothie de fruta	4.00
Gofres Belgas con fresas	7.95
Huevos revueltos con Tostadas	6.95
Tostada Francesa	4.50
Tostón con Salmón	6.95

Fig. 4. Resultado obtenido.

Vídeo 1. "Uso de <xsl:for-each> en Oxigen XML Editor"
https://bit.ly/2YTDgH9

/ 5. Caso práctico 1: "Transformar XML de notas"

Planteamiento: De cara a trabajar mejor con el fichero XML de notas, Alberto decide que es necesario transformarlo para generar otro XML.

Nudo: La idea que tiene Alberto es que haya una etiqueta 'alumno', y que contenga los atributos del ciclo, módulo y la nota. De esta manera, el fichero XML será más compacto.

Desenlace:

```
<alumno ciclo="DAW" modulo="LLMM" nota="9">Federico Ruiz</alumno>
<alumno ciclo="DAW" modulo="LLMM" nota="2">Silvia Martín</alumno>
<alumno ciclo="DAW" modulo="LLMM" nota="8">Manolo Chicón</alumno>

<alumno ciclo="DAW" modulo="PRO" nota="6">Federico Ruiz</alumno>
<alumno ciclo="DAW" modulo="PRO" nota="6">Silvia Martín</alumno>
<alumno ciclo="DAW" modulo="PRO" nota="4">Manolo Chicón</alumno>
<alumno ciclo="DAW" modulo="PRO" nota="2">Gloria Chicón</alumno>
```

Código 11. Resultado obtenido.

```
<?xml version="1.0"  encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:strip-space elements="instituto ciclos notas"/>
    <xsl:output method="xml"/>
    <xsl:template match="ciclos"/>
    <xsl:template match="//alumno">
                <alumno>
                    <xsl:attribute name="ciclo">
                        <xsl:value-of select="../@ciclo"/>
                    </xsl:attribute>
                    <xsl:attribute name="modulo">
                        <xsl:value-of select="../@nombre"/>
                    </xsl:attribute>
                    <xsl:attribute name="nota">
                        <xsl:value-of select="nota"/>
                    </xsl:attribute>
                    <xsl:value-of select="nombre"/>
                </alumno>
    </xsl:template>
</xsl:stylesheet>
```

Código 12. Fichero XSL.

/ 6. Elementos XSLT: <xsl:sort>

Esta instrucción permite ordenar los resultados obtenidos en la salida. Para ello, simplemente hay que incluirlo dentro de una instrucción <xsl:for-each>. La sintaxis es:

```
<xsl:sort select="expresion"
          order="ascending|descending"
          case-order="upper-first|lower-first"/>
```

Código 13. Sintaxis de <xsl:sort>.

Atributos:

- **select:** Expresión en formato *string* para indicar el nodo sobre el cual se realiza la ordenación.
- **order:** Para indicar si el orden es ascendente o descendente.
- **case-order:** Se indica si las mayúsculas van delante o detrás de contenidos en minúsculas.

```xml
<?xml version="1.0"  encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:template match="/carta">
        <html>
            <head>
                <style>
                    table{border:solid 2px black;
                    box-shadow:10px 10px 10px grey;}
                    th{color:white;background-color:#159dcc;}
                    td{border:solid 2px black;}
                </style>
            </head>
            <body>
                <table>
                    <th>Nombre</th>
                    <th>Precio</th>
                    <xsl:for-each select="desayuno">
                        <xsl:sort select="precio" order="descending"/>
                        <tr>
                            <td><xsl:value-of select="nombre"/></td>
                            <td><xsl:value-of select="precio"/></td>
                        </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Código 14. Uso de la ordenación en XSLT.

Nombre	Precio
Gofres Belgas con fresas	7.95
Huevos revueltos con Tostadas	6.95
Tostón con Salmón	6.95
Tostada Francesa	4.50
Smoothie de fruta	4.00

Fig. 5. Resultado obtenido.

Audio 1. "Otros atributos de <xsl:sort>"
https://bit.ly/3hYLF3u

/ 7. Herramientas de procesamiento

Como ya hemos comentado al principio del tema, un procesador XSLT permite leer un documento XML y aplicarle las instrucciones de la hoja de estilos XSLT para obtener el documento de salida. Uno de los procesadores que se pueden utilizar es el procesador integrado en el navegador. Los navegadores presentan diferencias y políticas de seguridad, por lo tanto, hay que estudiar en detalle su funcionamiento para que la conversión sea adecuada.

Existe *software* específico que nos va a permitir realizar el proceso de transformación. Caben destacar los siguientes:

- Oxygen XML Editor: *Software* comercial y uno de los más utilizados por sus numerosas herramientas que permiten crear, testear, validar, procesar cualquier documento XML.
- Dispone de una versión trial para poderlo valorar durante 30 días.
- Altova Editor XSLT: Este editor comercial tiene todos los aspectos relacionados con documentos XML, permitiendo el asistente de código, sangrado, marcadores, etc. También dispone de herramientas avanzadas que permiten expresiones XPath, aceleradores de velocidad y explorador integrado. También permite el procesamiento XSLT.
- Saxon: Es un procesador XSLT desarrollado por Saxonica Limited y dispone de varias versiones, siendo la home-edition una versión *open-source*.
- Xalan: Es un procesador XSLT desarrollado por Apache y permite transformar documentos XML en HTML, texto u otros documentos XML. Implementa XSLT versión 1.0 y permite expresiones XPath.
- Online XSLT Test Tool: Herramienta online que permite indicar el fichero XML, el fichero XSLT, y muestra el resultado de la transformación. Dispone también de algunos ejemplos para poder procesarlos.

Fig. 6. Herramienta online de procesamiento XSLT

```xml
<desayuno>
    <nombre>Smoothie de fruta</nombre>
    <precio>4.00</precio>
    <descripcion>Fresa, plátano y mango.</descripci
    <calorias>250</calorias>
</desayuno>
<desayuno>
    <nombre>Gobres Belgas con fresas</nombre>
    <precio>7.95</precio>
    <descripcion>Ligeros gofres belgas cubiertos de
    <calorias>900</calorias>
</desayuno>
<desayuno>
    <nombre>Huevos revueltos con Tostadas</nombre>
```

Código 15. Fichero XML

7.1. Depuración

El proceso de depuración permite seguir la generación del documento de salida a partir del documento XML aplicándole la hoja de estilo XSLT.

Los depuradores son muy útiles de cara a poder localizar posibles errores en nuestras instrucciones XSLT. Permiten acciones como ejecutar el código paso a paso, poner puntos de ruptura, ejecutar hasta donde se encuentre el cursor, pausar, detener, etc.

Fig. 7. Botones de depuración.

Oxygen XML Editor dispone de este tipo de depuradores.A continuación, se muestra la depuración con el código 9 del tema:

```
<body>
    <table>
        <th>Nombre</th>
        <th>Precio</th>
        <xsl:for-each select="desayuno">
            <xsl:if test="precio &lt; 5">
                <tr>
                    <td><xsl:value-of select="nombre"/></td>
                    <td><xsl:value-of select="precio"/></td>
                </tr>
            </xsl:if>
        </xsl:for-each>
    </table>
```

Código 16. Fichero XSLT.

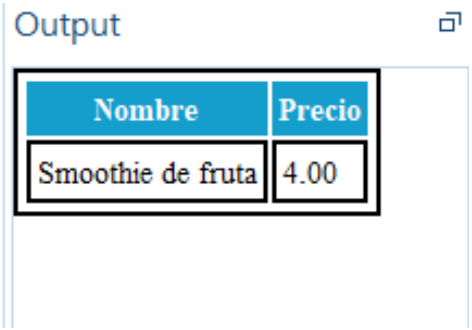

Fig. 8. Resultado parcial.

Vídeo 2. Uso de la depuración en Oxigen XML Editor"

https://bit.ly/31MayKo

A medida que vamos pulsando en el icono, se van ejecutando las instrucciones XSLT y se va marcando el nodo que se está procesando en XML y la instrucción asociada en el XSLT.

Al mismo tiempo, van apareciendo los resultados de cada una de esas instrucciones en la salida. En este caso, se ha ejecutado una iteración y se muestra el primer desayuno. Como el segundo desayuno no cumple la condición de que el precio sea menor de 5€, la instrucción saltará ese nodo y comprobará el siguiente.

/ 8. Caso práctico 2: "Listado de ciclos por especialidades"

Planteamiento: Son muchos los listados que necesita el instituto de formación, y en este caso, necesita un listado de los ciclos en formato HTML.

Nudo: Las condiciones que debe cumplir este listado son:

- Titulo con <h2> de las especialidades.
- Ciclos numerados.
- Los ciclos deben aparecer ordenados de manera ascendente.
- Solo deben aparecer los ciclos superiores.

Desenlace:

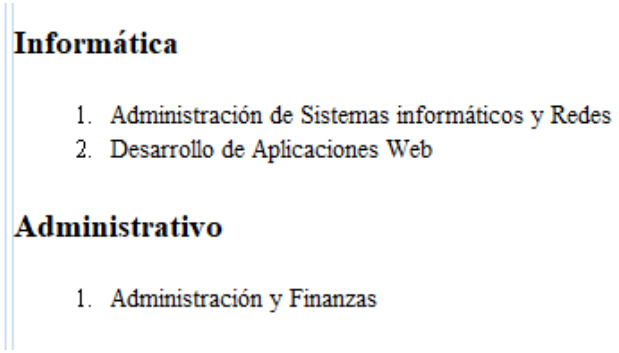

Fig. 9. Resultado obtenido.

```
<?xml version="1.0"  encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:strip-space elements="instituto ciclos notas"/>
    <xsl:output method="html"/>
    <xsl:template match="/">
        <html>
            <head><title>Ciclos</title></head>
            <body>
                <xsl:apply-templates/>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="//ciclos/especialidad">
        <h2><xsl:value-of select="@nombre"/></h2>
        <ol>
            <xsl:for-each select="ciclo">
                <xsl:sort select="."/>
                <xsl:if test="@grado='Superior'">
                   <li><xsl:value-of select="."/></li>
                </xsl:if>
            </xsl:for-each>
        </ol>
    </xsl:template>
    <xsl:template match="notas">
    </xsl:template>
</xsl:stylesheet>
```

Código17. Fichero XSL.

/ 9. XSL-FO

Hasta ahora, las transformaciones que se han estado realizando estaban orientadas a obtener un texto HTML u otro XML. Con XSL-FO (eXtensible StyleSheets Language Formating Objects), lo que se especifica es cómo queremos que se formateen los datos de origen XML para poderlos presentar en papel, impresora o en pantalla con un determinado formato. Entre estos formatos, destacan el formato PDF, SVG y RTF.

El proceso que habría que realizar se muestra en la siguiente figura:

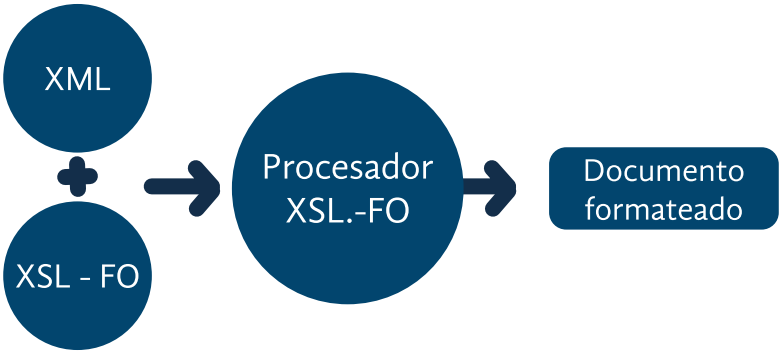

Fig. 10. Procesamiento XSL-FO.

La unidad básica en un documento XSL-FO es el objeto que puede representar páginas, párrafos, textos, etc.

Por lo tanto, todas las etiquetas específicas del lenguaje XSL-FO están relacionadas con elementos de maquetado. Un listado completo de estas etiquetas y cómo utilizarlas las podemos consultar en el siguiente enlace.

A modo de ejemplo, se va a realizar una transformación de la carta de desayunos para obtener un resultado en PDF, tal y como se muestra en la siguiente figura. Se ha usado *Oxygen XML Editor.*

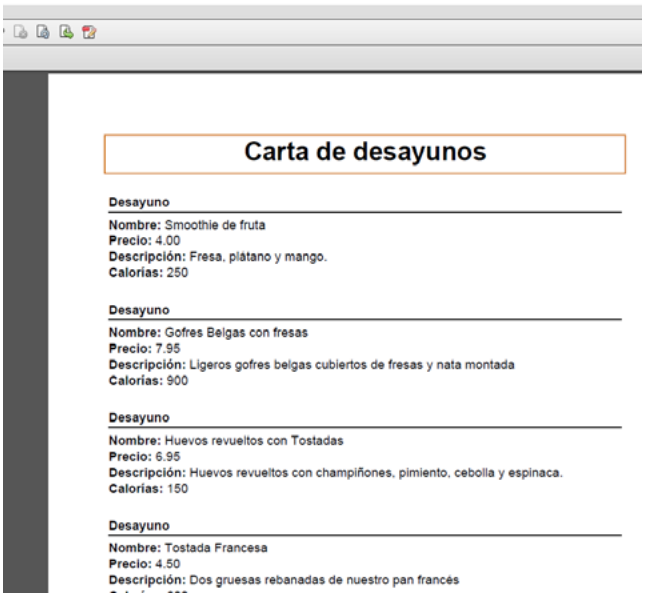

Fig. 11. Documento PDF generado

El código completo para generar este documento lo podemos consultar en el **anexo** del tema.

/ 10. Resumen y resolución del caso práctico de la unidad

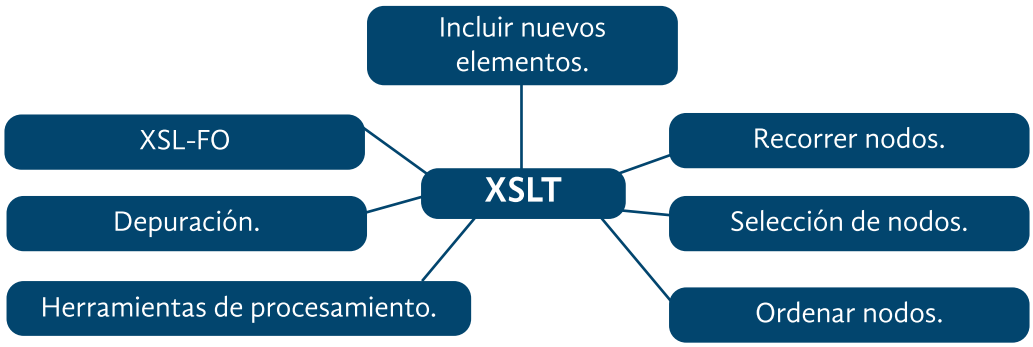

Fig. 12. Esquema del tema

En este tema, se han completado los **elementos** que se pueden utilizar como instrucciones en una transformación XSLT. Entre estos elementos, destacan la inclusión de nuevos elementos con <xsl:element> y <xsl:attribute>, el recorrer los nodos con <xsl:for-each>, seleccionar ciertos nodos con <xsl:if> o <xsl:choose>, y, por último, poder ordenar los nodos con <xsl:sort>.

También, se han listado **herramientas de procesamiento** y lo importante que es disponer de la **depuración** para poder identificar posibles errores en el XSLT. Para terminar, se ha hecho una breve introducción a **XSL-FO.**

Aquí disponemos de un proyecto donde podemos probar todos los ejemplos vistos en el tema, experimentar con otros nuevos y visualizar los resultados en texto o HTML.

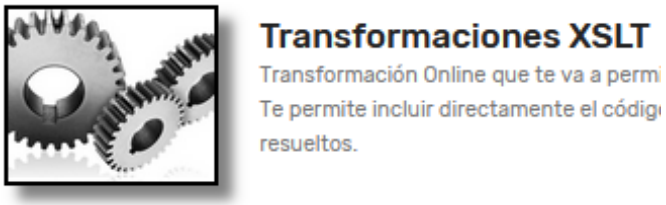

Fig. 13 Transformación online en Webmalagadesign

Resolución del caso práctico de la unidad

Málaga Formación

Notas Alumnos

Nombre	Nota	Módulo
Federico Ruiz	Sobresaliente	LLMM
Manolo Chicón	Notable	LLMM
Federico Ruiz	Bien	PRO
Silvia Martín	Bien	PRO
Manolo Chicón	Suspenso	PRO
Silvia Martín	Suspenso	LLMM
Gloria Chicón	Suspenso	PRO

Fig. 14. Notas de los alumnos.

Como planteamos al principio, ahora,el instituto necesita ordenar en forma de tabla las notas de los alumnos, destacando aprobados y suspensos. Aquí podemos ver una representación de cómo podría quedar. El XML con las notas y el fichero XSL lo podemos consultar en el **anexo** del tema.

/ 11. Webgrafía

XSL-FO: Referencia de XSL-FO

Herramientas de procesamiento y depuración

Oxygen XML Editor
Altova Editor XSLT
Saxon
Xalan
Online XSLT Test Tool
Referencia XSLT en W3Schools
Recomendación XSLT de W3C