

# TRANSFORMACIÓN DE UN ARCHIVO XML MEDIANTE UNA HOJA DE ESTILO XSLT PRODUCTO 5

Daniel Carrasco Luque

FP.047 – (P) Programa comandos personalizados para el sistema operativo

## ÍNDICE

Introducción.....	2
Plantilla HTML .....	2
Desarrollo plantilla XSLT .....	5
Mejoras para hacer más atractivo el HTML .....	7
Aspecto HTML resultante .....	9
Webgrafía .....	9

## Introducción

Para este producto se nos solicita crear una hoja de estilo XSLT con la capacidad de transformar el fichero XML obtenido en el producto 4 de este proyecto. Para la obtención del fichero XSL resultante se ha seguido la estrategia de desarrollo propuesta en los vídeos tutoriales proporcionados en el apartado “recursos de aprendizaje”, de los cuales dejo el enlace en el apartado de webgrafía.

Primeramente, he desarrollado una plantilla básica en HTML la cual me da la estructura para presentar los datos, posteriormente sobre la plantilla HTML hemos implementado las etiquetas XSL, estas permitirán obtener los datos del fichero XML, tratarlos si es necesario, y presentarlos en el fichero HTML final.

## Plantilla HTML

Para la creación de la plantilla se ha investigado cuales son las etiquetas HTML básicas para conformar la estructura del fichero y así comprender cual es la forma de trabajar de estos ficheros.

Etiquetas:

- **<html>** : Etiqueta raíz del documento HTML.
- **<head>** : Cabecera del documento. Contiene metadatos, el título del documento mediante <title>, enlaces a archivos CSS, scripts, etc. Esta parte no es visible.
- **<body>** : Contiene el contenido visible de la página web. El contenido de esta etiqueta es el que debemos desarrollar para mostrar los datos del fichero XML.

Si desarrollo las etiquetas anteriores en un fichero .html ya tendremos la base para comenzar a construir la visualización de lo que queramos mostrar.



```

1  <html>
2    <head>
3      <title>Adaptador de Red</title>
4    </head>
5    <body>
6      <h1>Adaptador de Red</h1>
7    </body>
8  </html>

```



Si observamos las 2 imágenes anteriores podemos ver que se ha desarrollado dentro de la etiqueta <head> una etiqueta <title> la cual muestra en la pestaña del navegador el texto indicado. En <body> hemos usado una etiqueta <h1> la cual le da un formato predefinido al texto contenido. Existen 6 etiquetas que van de <h1> a <h6>, de mayor tamaño de texto a menor y se usan para indicar encabezados o títulos.

Una vez tenemos la base se ha buscado la forma más adecuada para representar los datos de nuestro adaptador, para ello se a desarrollar en forma de tabla ya que considero que es la forma óptima de representar nuestros datos.

Para definir una tabla tenemos las siguientes etiquetas:

- **<table>**: Esta etiqueta define el contenedor de la tabla.
- **<tr>**: define una fila en la tabla, “table row”.
- **<th>**: define la celda de encabezado en la tabla. Tiene la particularidad de mostrar el texto contenido en negrita sin necesidad de especificarlo, se suele usar en la primera fila, “table head”.
- **<td>**: Define una celda para datos en la tabla, “table data”.

Desarrollando las etiquetas mostradas para nuestro fichero XML obtendremos la siguiente codificación y su visualización en el navegador web.

Para obtener este resultado se ha desarrollado una etiqueta <tr> para cada fila, cada fila contiene una etiqueta <th> para la columna de la izquierda y una etiqueta <td> para la columna de la derecha.

Para poder mostrar las líneas de las tablas se ha hecho uso del atributo border=”1” de la siguiente forma sobre la etiqueta <table border=”1”>



<b>Nombre</b>	#Nombre_adaptador#
<b>IP Adaptador</b>	#IP#
<b>Mascara</b>	#Mascara_de_subred#
<b>Tipo DNS</b>	#tipo_DNS#
<b>IP DNS</b>	#Valor_DNS#
<b>Tiempo respuesta</b>	#tiempo_ms#
<b>Número Saltos</b>	#Cantidad_saltos#
<b>IP Saltos</b>	#IP_de_saltos#

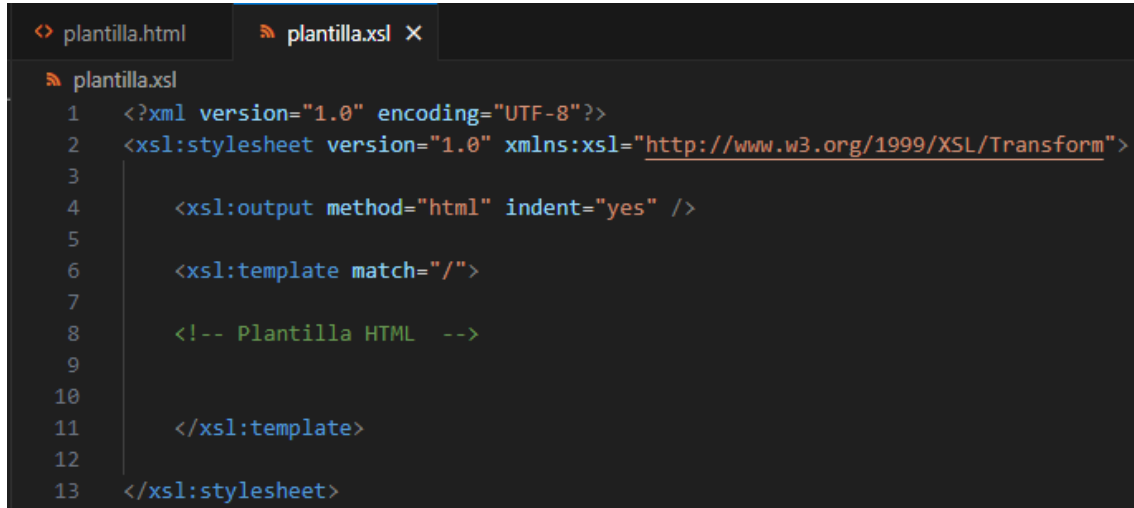
```

<> plantilla.html X
<> plantilla.html > ...
1  <html>
2      <head>
3          <title>Adaptador de Red</title>
4      </head>
5      <body>
6          <h1>Adaptador de Red</h1>
7          <table border="1">
8              <tr>
9                  <th>Nombre</th>
10                 <td>#Nombre_adaptador#</td>
11             </tr>
12             <tr>
13                 <th>IP Adaptador</th>
14                 <td>#IP#</td>
15             </tr>
16             <tr>
17                 <th>Mascara</th>
18                 <td>#Mascara_de_subred#</td>
19             </tr>
20             <tr>
21                 <th>Tipo DNS</th>
22                 <td>#tipo_DNS#</td>
23             </tr>
24             <tr>
25                 <th>IP DNS</th>
26                 <td>#Valor_DNS#</td>
27             </tr>
28             <tr>
29                 <th>Tiempo Respuesta </th>
30                 <td>#tiempo_ms#</td>
31             </tr>
32             <tr>
33                 <th>Número Saltos</th>
34                 <td>#Cantidad_saltos#</td>
35             </tr>
36             <tr>
37                 <th>IP Saltos</th>
38                 <td>#IP_de_saltos#</td>
39             </tr>
40         </table>
41     </body>
42 </html>

```

## Desarrollo plantilla XSLT

La plantilla HTML creada anteriormente se debe acomodar entre los comandos básicos XSL que transformarán nuestro código HTML en un fichero XSLT funcional. Para ello el código HTML debe quedar contenido dentro de las siguientes etiquetas:



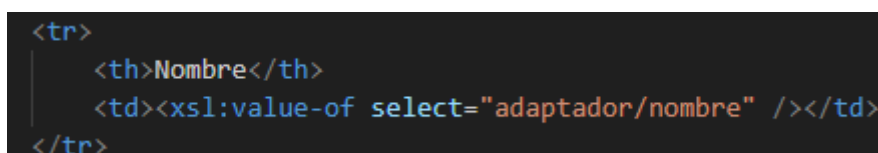
```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3
4      <xsl:output method="html" indent="yes" />
5
6      <xsl:template match="/">
7
8          <!-- Plantilla HTML -->
9
10
11      </xsl:template>
12
13 </xsl:stylesheet>
  
```

- Haciendo uso de la primera etiqueta declaramos la versión del fichero XML y el tipo de codificación del texto contenido tal y como ya hicimos para el documento XML resultante del producto 4.
- La etiqueta **<xsl:stylesheet>** declara que el fichero codifica una hoja de estilo XSL, indicando su versión y declarando el estándar seguido.
- La etiqueta **<xsl:template match="/">** establece que la creación y aplicación de la plantilla que contenga, accederá a los datos del fichero fuente desde el nodo raíz "/", es decir la etiqueta más general de nuestro fichero XML, <adaptador>.
- La sentencia **<xsl:output>** enmarcada en el cuadro rojo no es esencial para generar nuestro fichero de salida pero considero que es una buena práctica aplicarlo. Una hoja de estilo XSLT puede generar múltiples tipos de fichero de salida como texto plano, HTML o incluso otros ficheros XML. El campo "indent="yes"" fuerza que el documento HTML resultante esté indentado y sea más fácil de comprender por un posible lector.

Una vez hemos introducido la estructura HTML dentro de nuestro nuevo fichero XSL ya estamos listos para, mediante etiquetas XSL, extraer los datos del fichero XML e insertarlos en los puntos que definimos en la plantilla inicial HTML.

Para lograr lo mencionado anteriormente se utiliza la etiqueta **<xsl:value-of select="nombre\_ruta">** donde "select" tendrá como parámetro la ruta de etiquetas del fichero XML para obtener el valor indicado. Ejemplo:



```

<tr>
  <th>Nombre</th>
  <td><xsl:value-of select="adaptador/nombre" /></td>
</tr>
  
```

## Adaptador de Red

Nombre	Wi-Fi
--------	-------

Esta forma de proceder se ha implementado para el resto de las filas que queremos representar en nuestro documento HTML final excepto para la última fila. La última fila se ha implementado con un bucle “for-each” ya que esta fila puede contener un número indeterminado de posibles campos a representar, se trata de las ips registradas para cada salto. Este punto, como ya indiqué para el producto 4 le puse especial atención ya que mi implementación inicial usaba etiquetas indexadas para cada salto <salto1>...<saltoX> y el avanzar a investigar sobre el producto 5 me ayudó a definir todas las ip de salto con la misma etiqueta <ip>.

Para implementar el bucle simplemente debemos implementar la etiqueta **<xsl:for-each select=“Nombre\_ruta”>** seguido de las instrucciones que sean pertinentes y posteriormente cerrar el bucle con **</xsl:for-each>**.

```
<tr>
  <th>IP Saltos</th>
  <td>
    <xsl:for-each select="adaptador/DNS/saltar/ip">
      <xsl:value-of select="." />
      <br>
    </xsl:for-each>
  </td>
</tr>
```

Si vemos el contenido que hay entre las 2 etiquetas “for-each” se usa un “value-of” y una etiqueta <br>, vamos a explicar el por que:

- **<xsl:value-of select="." />**: Como ya hemos visto “value-of” nos permite obtener el valor que debemos extraer del fichero XML, en este caso el select hace referencia a un punto, esto implica que se debe usar el valor que tengamos actualmente seleccionado, este valor ya seleccionado nos lo está dando el “for-each” anterior por lo cual solamente debemos darle el tratamiento de mostrarlo en el fichero resultante.
- **<br>**: “break”, esta etiqueta simplemente introduce un salto de línea. Es una de las pocas etiquetas vacías que existen en HTML por lo cual no requiere de cierre con </br>. Se ha hecho uso de ella para poder mostrar las ips en columna, de no ser así las ips se mostrarían en forma de lista concatenada.

**Nota:** Posteriormente cuando he procesado los ficheros .xml y .xsl para obtener el fichero .html he tenido problemas con la etiqueta <br>. Al procesar el fichero con “Saxon” no admite una etiqueta abierta, en la versión definitiva se ha cambiado por </br>. Al trabajar con el editor web proporcionado en los recursos esto no sucedía.

## Adaptador de Red

<b>Nombre</b>	Wi-Fi
<b>IP Adaptador</b>	192.168.1.13
<b>Mascara</b>	255.255.255.0
<b>Tipo DNS</b>	primaria
<b>IP DNS</b>	1.1.1.1
<b>Tiempo Respuesta</b>	8
<b>Número Saltos</b>	9
<b>IP Saltos</b>	192.168.1.1 Desconocida Desconocida 10.255.200.65 10.34.201.129 10.34.65.173 81.52.188.113 193.251.150.10 1.1.1.1

### Mejoras para hacer más atractivo el HTML

Una vez realizados los pasos anteriores obtenemos una tabla que muestra de forma correcta y ordenada nuestros datos del fichero HTML pero, he querido ir un poco más allá he investigado un poco más para implementar mejoras sobre nuestro resultado.

- Lo primero que he hecho ha sido cambiar el fondo blanco de la página por otro color aplicando un poco de CSS con una etiqueta `<style>` alojada dentro de `<head>`

```

8      <head>
9          <title>Adaptador de Red</title>
10         <style>
11             body {
12                 background-color: #BDF1CA;
13             }
14         </style>
15     </head>
```

- Todos los campos que he implementado en etiquetas `<th>` aparecen siempre en negritas y centrados, he buscado la forma de alinearlos a la izquierda implementando atributos `"align="left"` para todos los campos. Para el campo que muestra "IP Saltos" he usado `"align="left"` y `"valign="top"` de esta forma puedo alinear el texto a la izquierda y arriba de la casilla de la tabla.

```
<th align="left">Nombre:</th>
```

```
<th align="left" valign="top">IP Saltos:</th>
```



- Me ha parecido correcto introducir la unidad de tiempo en la casilla que se muestra el tiempo de respuesta, para ello se ha introducido una nueva etiqueta `<xsl:text>` la cual permite añadir el texto que creas oportuno, en mi caso “ms”.

```
<td>
  <xsl:value-of select="adaptador/DNS/tiempo" />
  <xsl:text> ms</xsl:text>
</td>
```

- Se ha implementado que los campos dedicados al nombre del adaptador las casillas aparezcan de un color más amarillo y el nombre en azul y negrita. Para ello se hace uso de un atributo “style” donde “background-color” define el color de las casillas proporcionado en valor hexadecimal, “color: blue” establece el texto de color azul y “font-weight: bold” establece el texto en negritas.

```
<tr style="background-color: #AAFFAA; color: blue; font-weight: bold;">
  <th align="left">Nombre:</th>
  <td><xsl:value-of select="adaptador/nombre" /></td>
</tr>
```

- Como última mejora me ha parecido interesante saber si era posible implementar diferentes formas de mostrar un texto en caso de que se cumplan determinadas condiciones, lo cual me ha permitido implementar que cada vez que aparezca una ip marcada como “Desconocida” en la lista de saltos la palabra desconocida se muestre en negrita y de color rojo. Esto es posible realizarlo mediante sentencias XML.

```
<tr>
  <th align="left" valign="top">IP Saltos:</th>
  <td>
    <xsl:for-each select="adaptador/DNS/saltar/ip">
      <xsl:choose>
        <xsl:when test=". = 'Desconocida'">
          <span style="color: red; font-weight: bold;">Desconocida</span>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="." />
        </xsl:otherwise>
      </xsl:choose>
      <br/>
    </xsl:for-each>
  </td>
</tr>
```

La etiqueta `<xsl:choose>` nos permite crear una estructura de control al estilo “if-else” o “switch” para posteriormente mediante etiquetas `<xsl:when>` analizar que se cumpla la condición. Se pueden tener múltiples “when” lo que se debe tener en cuenta es que siempre se ejecutará el primero que se cumpla. Con el atributo “test” se establece la condición a cumplir. En caso de no cumplirse ninguna condición evaluada en “when” se hace uso de `<xsl:otherwise>` el cual permite definir la acción por defecto en caso de que no se cumpla ninguna previa en “when”. De esta forma

he podido evaluar si existen ips con valor 'Desconocida' y aplicarle el formato definido.

La etiqueta <span> forma parte del lenguaje HTML y se ha combinado con sintaxis de CSS mediante "style" para dar el formato deseado. "Span" se puede introducir dentro de cualquier texto sin modificar las propiedades del resto del texto o párrafo donde se contenga.

## Aspecto HTML resultante

Adaptador de Red	
Nombre:	Wi-Fi
IP Adaptador:	192.168.1.13
Mascara:	255.255.255.0
Tipo DNS:	primaria
IP DNS:	1.1.1.1
Tiempo Respuesta:	8 ms
Número Saltos:	9
IP Saltos:	192.168.1.1 Desconocida Desconocida 10.255.200.65 10.34.201.129 10.34.65.173 81.52.188.113 193.251.150.10 1.1.1.1

Para visualizar el documento HTML resultante se ha usado [XSLT Tryit Editor v1.2](#) proporcionado en los recursos del aula.

## Webgrafía

- [Introducción al lenguaje de transformación XSL - YouTube](#)
- [XSLT Tryit Editor v1.2](#)
- [HTML Tutorial](#)
- [HTML Styles CSS](#)
- [XSLT <xsl:choose> Element](#)
- [XSLT Introduction](#)