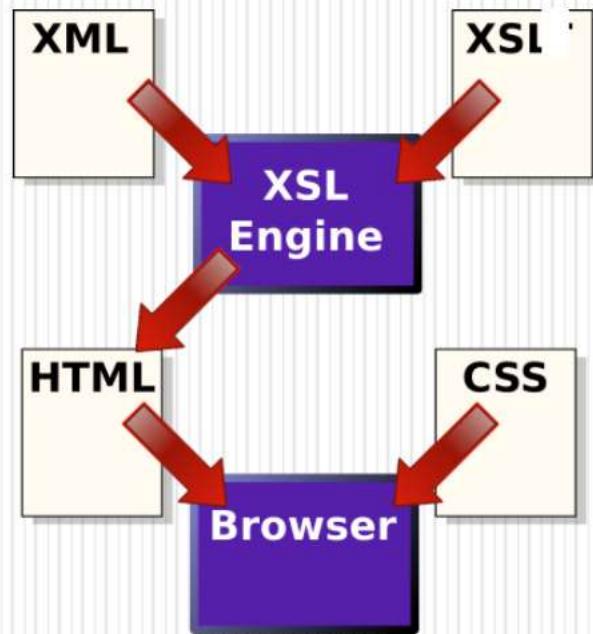


Unidad 4: XSLT y XPATH



JJ Taboada León
IES San Sebastián, Departamento de Informática
LENGUAJE DE MARCAS Y SGI
Curso 2011 / 2012

Guión del tema

- ¿Qué es XSLT?
- Aplicación de las transformaciones
- Estructura y sintaxis XSLT
- Elementos XSLT
- XSLT con CSS

EJERCICIOS

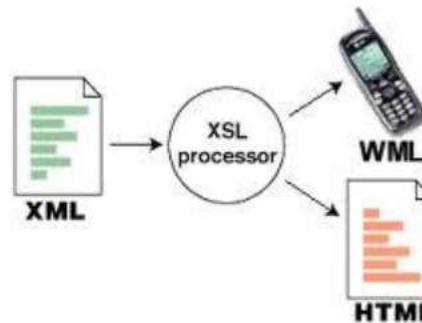
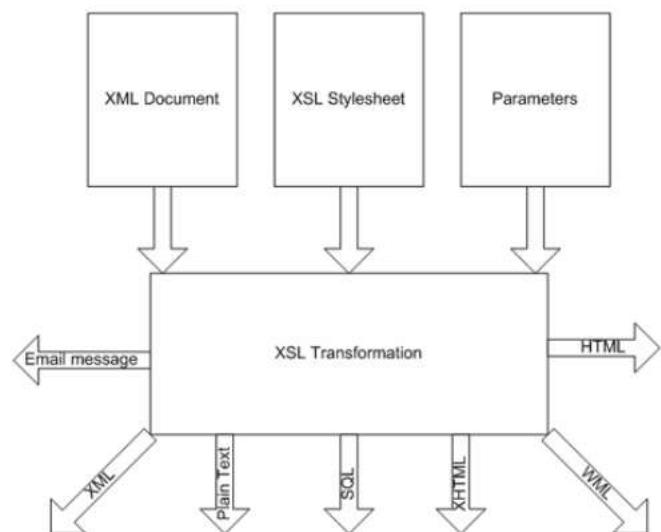
- Transformar documentos XML con Notepad++ y Dreamweaver
- Prácticas con XML HTML y CSS

¿QUÉ ES XSLT?

- XSLT (eXtensible Stylesheet Language – Transformations),
Lenguaje de hoja de estilo ampliable para transformaciones
 - Describe un lenguaje basado en XML para transformar documentos XML a cualquier otro formato
- XSLT es el lenguaje de hojas de estilo recomendado de XML.
- XSLT es mucho más sofisticado que el CSS.
- XSLT puede ser utilizado para transformar documentos XML en HTML, antes de ser mostrados en un navegador.

Aplicación de las transformaciones

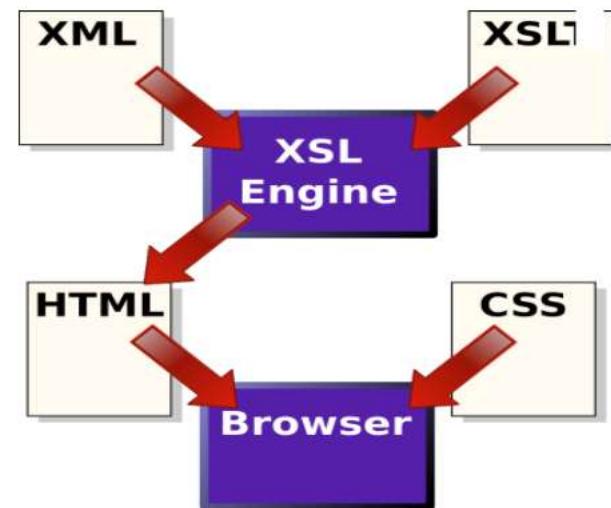
- Utilizaremos XSLT para transformar documentos entre esquemas XML que permitan su procesamiento por distintos sistemas
- Utilizaremos XSLT para transformar documentos XML en HTML, WML, o cualquier otro formato que facilite su presentación en la pantalla de un ordenador o en impresora
- La transformación de XML a HTML es el principal uso que se hace de XSLT



WML (Wireless Markup Language)
lenguaje que se utiliza para construir las páginas que aparecen en las pantallas de los teléfonos móviles y los asistentes personales digitales ([PDA](#))

Aplicación de las transformaciones

- No debemos confundir las transformaciones XSLT con la presentación de documentos XML con CSS
- Con XSLT, generaremos un documento HTML a partir de un documento XML. Se tratará de *dos documentos “distintos”*
- Con CSS, el navegador recibe un documento XML que formatea utilizando las reglas CSS para presentarlo en pantalla de forma que sea más fácilmente legible, pero es el mismo documento



HERRAMIENTAS

- Actualmente contamos con varias herramientas para realizar transformaciones XSLT:
 - Saxon, desarrollado en Java por Michael Kay (un gurú de XSLT)
 - xt, diseñado por James Clark
 - Dreamweaver
 - XMLspy
 - En las prácticas usaremos Notepad++ con el complemento XMLtools y Dreamwaver

Estructura de una hoja de estilo XSLT

- Una hoja de estilo XSLT es un documento XML. Debe estar bien formado.
- Las hojas de estilo se guardarán siempre en archivos independientes con extensión **.xsl**
- Deben comenzar con una declaración XML:
`<?xml version="1.0"?>`
- El elemento raíz de la hoja de estilo XSLT es **stylesheet**.
- 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.

Estructura de una hoja de estilo XSL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

.....
.....
</xsl:stylesheet>
```

Estructura de una hoja de estilo XSL

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

Estructura de una hoja de estilo XSL

REGLAS DE TRANSFORMACIÓN

EJEMPLO:

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

```
  <html>
    <body>
      <h2> <xsl:value-of select=". " /> </h2>
    </body>
  </html>
</xsl:template>
```

árbol de origen al que se aplica esta plantilla

} salida

- La regla se aplicará a todas las instancias del elemento nombre. Esto se indica mediante el atributo **match** que acompaña al elemento **xsl:template**.
- Entre las etiquetas de inicio y de fin del elemento **xsl:template** se escribe la transformación que se debe realizar...
- es decir, **qué texto y qué marcas se escribirán en el documento resultado de la transformación**, cada vez que se encuentre una instancia del elemento **nombre** en el documento origen.
- Con **<xsl:value-of...>**, se recupera y escribe en el documento resultado el valor del elemento que está siendo procesado.

Ejemplo transformación XSLT

Documento XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xmlstylesheet type="text/xsl" href="prueba.xsl" ?>
<ciudades>
    <ciudad>
        <nombre>Madrid</nombre>
        <habitantes>3500000</habitantes>
    </ciudad>
    <ciudad>
        <nombre>Málaga</nombre>
        <habitantes>800000</habitantes>
    </ciudad>
    <ciudad>
        <nombre>Toledo</nombre>
        <habitantes>50000</habitantes>
    </ciudad>
</ciudades>
```

The diagram illustrates the relationship between an XML document and an XSLT stylesheet. On the left, the XML document is shown in red text. Two arrows point from the XML code to two gray boxes on the right. One arrow points from the line '<?xmlstylesheet type="text/xsl" href="prueba.xsl" ?>' to a box labeled 'Referencia al documento XSL'. Another arrow points from the opening '<ciudades>' tag to a box labeled 'Documento XML'.

Referencia al documento XSL

Documento XML

Ejemplo transformación XSLT

Documento XSL

```
<?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="//nombre">
    <html>
      <body>
        <h2> <xsl:value-of select="." /> </h2>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Documento XSL

Queremos obtener los
nombres de las ciudades

Ejemplo transformación XSLT

Resultado

```
<h2>Madrid</h2>3500000  
<h2>Málaga</h2>800000  
<h2>Toledo</h2>50000
```

- El resultado obtenido no es el documento HTML esperado
- Vemos que en el documento de salida no sólo se ha incluido el texto de los elementos procesados, sino el de todos los elementos del documento original...
- Para evitar ésto, tenemos que hacer unos cambios en la hoja de estilo XSLT (ver siguiente página):

Ejemplo transformación XSLT

Documento xsl mejorado

```
<?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>
            <head>
                <title>Ejemplo XSLT</title>
            </head>
            <body>
                <h1> CIUDADES DE ESPAÑA </h1>

                <xsl:apply-templates select="//nombre" />

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

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

</xsl:stylesheet>
```

La regla “de inicio”

- La regla `<xsl:template match="/">` se ejecuta cuando se encuentra el elemento raíz del documento XML
- Dentro de esta regla, podemos incluir llamadas a otras reglas definidas en la hoja de estilo, mediante el elemento:
`<xsl:apply-templates select="..." />`
- El atributo `select` 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

La regla “de inicio”

- El resultado de la transformación es el siguiente:

```
<html>
  <head>
    <title>Ejemplo XSLT</title>
  </head>
  <body>
    <h1> CIUDADES DE ESPAÑA </h1>
    <h3>Madrid</h3>
    <h3>Málaga</h3>
    <h3>Toledo</h3>
  </body>
</html>
```

Ejemplos

simple.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xmlstylesheet type="text/xsl" href="simple.xsl"?>
<simple>
    <nombre> Álvarez Martín Almudena </nombre>
    <nombre> Aragon Batista Joaquin </nombre>
    <nombre> Chavarri Satalaya Jordan </nombre>
    <nombre> Cruz Florencio Victor </nombre>
    <nombre> Del Toro del Toro Moisés </nombre>
    <nombre> Díaz Redondo Raul </nombre>
</simple>
```

Simple.xsl

```
<?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>
            <head>
                <title>Ejemplo XSLT simple</title>
            </head>
            <body>
                <xsl:apply-templates />

                </body>
            </html>
        </xsl:template>
        <xsl:template match="nombre">
            <p> Nombre encontrado</p>
        </xsl:template>

    </xsl:stylesheet>
```

Actividades

1. Con Notepad++ escribir cada uno de los ejemplos anteriores y hacer todas las pruebas que se os ocurran.
2. Con Notepad++, crear una primera hoja de estilo XSLT que, a partir del documento XML [tienda1.xml](#), extraiga en un documento HTML los artículos de la tienda
3. Formatear la lista de artículos para que se presente como una lista no ordenada (sin numerar), de HTML
4. Cambiar la hoja de estilo XSLT para que los artículos se presenten en una tabla con una única columna
5. Cambiar la hoja de estilo para que los artículos se presenten en una tabla con dos columnas. En la primera de ellas se escribirá un texto fijo: “ARTICULOS DE MI TIENDA”
6. Aplicar esta última hoja de estilo al documentos XML [tiendecilla.xml](#)

El elemento <xsl:value-of...> (I)

- En el elemento <xsl:value-of...> se puede indicar que se quiere mostrar el valor del elemento que estamos procesando
- También podemos indicar que queremos mostrar el valor de un elemento hijo, o descendiente, del elemento que se está procesando
- En el ejemplo anterior, podríamos utilizar xsl:value-of para mostrar en el documento resultado de la transformación el artículo, la marca y el modelo de cada producto.
- Esto es posible porque en el atributo select podemos utilizar una “**expresión XPATH**”
- XPATH es un lenguaje que nos permite **direccinarnos a las secciones de un documento XML y obtener las partes de la información que deseamos** (nodo de contexto)

El elemento <xsl:value-of...> (II)

- Por ejemplo, para mostrar el valor del elemento articulo, que es un hijo del elemento producto, podríamos utilizar la siguiente regla:

```
<xsl:template match="//producto">  
    <xsl:value-of select=".//articulo" />  
</xsl:template>
```

El valor del atributo select se puede leer de la siguiente forma: “dame el valor del elemento articulo que es hijo del elemento que estoy procesando”. En este caso, cada uno de los elementos producto

Esto se indica mediante ./

Selección de nodos

Expresión	Descripción
<i>nodename</i>	Selecciona todos los nodos secundarios del nodo denominado
/	Selecciona desde el nodo raíz
//	Selecciona nodos en el documento desde el nodo actual que coincidan con la selección sin importar donde se encuentren
.	Selecciona el nodo actual
..	Selecciona el padre del nodo actual
@	Selecciona los atributos

Actividades

- Utilizando el documento [tienda1.xml](#), crear una hoja XSLT que transforme el documento xml en un documento HTML con la siguientes especificaciones:
 - El documento HTML deberá mostrar una tabla. La tabla contendrá una fila para cada producto.
 - Las filas tendrán tres celdas, en la que aparecerá el código, el artículo y la cantidad.

Resumen

- En las reglas XSLT, entre sus marcas de inicio y de fin, se puede incluir:
 - Texto que se escribirá “tal cual” en el documento resultado de la transformación.
 - Marcas HTML o XML que se añadirán al documento resultado de la transformación.
 - Elementos reservados de la especificación XSLT que realizarán una acción como recuperar el valor de un elemento, ordenar los resultados, llamar a otras reglas de la hoja de estilo, etc (llamadas **expresiones XPATH**).

Procesamiento de nodos por lotes

<xsl:for-each>

- El elemento <xsl:for-each> permite hacer un bucle en XSLT
 - se puede utilizar para seleccionar todos los elementos XML del nodo actual
 - **Actividad**
 - Aplicarlo en tiendecilla

```
<?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>
<h2>CIUDADES DE ESPAÑA</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Ciudades</th>
<th>Habitantes</th>
</tr>
<xsl:for-each select="ciudades/ciudad">
<tr>
<td><xsl:value-of select="nombre"/></td>
<td><xsl:value-of
select="habitantes"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

Orden de procesamiento

- Las reglas se van activando y ejecutando a medida que se recorre el documento origen que se quiere transformar.
- De esta forma, las reglas se ejecutan en el orden en el que se van encontrando los elementos en el documento.
- Este comportamiento por defecto puede cambiarse en las hojas de estilo XSLT, a diferencia de lo que sucedía en las hojas de estilo CSS
- Esto permite “reordenar” los contenidos del documento XML, de una forma distinta a como están ordenadas en el documento XML inicial

Orden de procesamiento

- Para ordenar los contenidos, se utiliza el elemento xsl:sort
- xsl:sort es un elemento hijo de xsl:apply-templates
- Acepta dos atributos:
 - select – que toma como valor el nombre del elemento que se va a utilizar como criterio de ordenación y
 - order – que indica si se debe utilizar un orden ascendente o descendente.

```
<xsl:apply-templates select="//ciudad">
    <xsl:sort select="ciudad" order="descending" />
</xsl:apply-templates>
```

- **ACTIVIDADES**
 1. **Modificar la hoja xsl del último ejercicio (salida en tres columnas) para que los artículos se ordenen en orden descendente.**
 2. **Aplicar la hoja de estilo XSL a tiendecilla, pero ordenado por cantidad.(especificar data-type)**

Asociar una hoja de estilo XSL a un documento

- Debemos incluir, tras la declaración XML, la siguiente instrucción de procesamiento:

```
<?xml-stylesheet type="text/xsl" href="hojaEstilo.xsl"?>
```

- Ejemplo

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xsl"  
    href="http://www.anaya.es/docs/xml/ejemplo.xsl"?>
```

```
<documento>
```

```
    <titulo>Programar ASP</titulo>
```

```
    <paginas>456</paginas>
```

```
    <anno-pub>2001</anno-pub>
```

```
</documento>
```

Leer y obtener el valor de atributos en XSLT

- En XSLT podemos “filtrar” o indicar qué instancias de un elemento queremos procesar, tomando como criterio de selección el valor de los atributos que acompañan a los elementos
- Para hacer esto, en un elemento `xsl:value-of`, podemos recuperar el valor de un atributo mediante la expresión **`@nombreAtributo`**, por ejemplo:

```
<libro genero="novela">
  <autor>Deepak Chopra</autor>
  <titulo>El sendero del Mago</titulo>
  <isbn>950-15-1727</isbn>
  <editorial>Harmany Book</editorial>
  <sumario>... nos muestra cómo debemos ... Por medio de
  historias como </sumario>
  <precio moneda="euros">50.00</precio>
</libro>
```

```
<xsl:template match="libro">
  <tr>
    <td><xsl:value-of select="@genero" /></td>
    <td><xsl:value-of select=".//precio/@moneda" /></td>
  </tr>
</xsl:template>
```

- ACTIVIDADES: Diseñar una hoja de estilo para `tiendecilla.xml` que presente los datos en una tabla donde aparezcan el artículo, la cantidad y el precio junto con la moneda de cada producto.

Elemento xsl:if

- El elemento <xsl:if> contiene una plantilla que se aplicará sólo si la condición especificada es verdadera.

SINTAXIS

```
<xsl:if test="expresion">  
    <!-- Contenido de template --&gt;<br/></xsl:if>
```

OPERADORES XSL	
Igualdad (=)	=
Desigualdad (≠)	!=
Menor que (<)	<
Mayor que (>)	>

EJEMPLO

```
<xsl:for-each select="//ciudad">  
  
    <xsl:if test="habitantes > 100000">  
        <h3> <xsl:value-of select=".//nombre" /> </h3>  
        <xsl:value-of select=".//habitantes" />  
    </xsl:if>  
  
</xsl:for-each>
```

Elemento xsl:choose

- El elemento <xsl:choose> se utiliza en conjunción con <xsl:when> y <xsl:otherwise> para expresar múltiples pruebas condicionales.

SINTAXIS

```
<xsl:choose>
    <xsl:when test="expression">
        ... Hacer algo...
    </xsl:when>
    ...
    <xsl:otherwise>
        ... Hacer algo ....
    </xsl:otherwise>
</xsl:choose>
```

Ejemplo de xsl:choose

EJEMPLO

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

  <xsl:choose>
    <xsl:when test="habitantes < 100000">
      <tr bgcolor="#ff0000">
        <td> <xsl:value-of select=".//nombre" /> </td>
        <td><xsl:value-of select=".//habitantes" /></td>
      </tr>
    </xsl:when>
    <xsl:otherwise>
      <tr bgcolor="#00ff00">
        <td> <xsl:value-of select=".//nombre" /> </td>
        <td> <xsl:value-of select=".//habitantes" /> </td>
      </tr>
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>
```

Actividades

1. Añadir a tiendecilla.xml dos productos nuevos. Sus precios serán 100 € y 350 € respectivamente.
2. Diseñar una hoja de estilos xsl para que los artículos con precio menores o iguales que 100 € aparezcan en color verde, los artículos con precios >100 € y < 1000 € de color azul y el resto rojo.

RESUMEN ELEMENTOS XSLT (1)

<u>xsl:apply-imports</u>	Invoca una regla de plantilla reemplazada.
<u>xsl:apply-templates</u>	Dirige el procesador XSLT para que busque la plantilla adecuada que se debe aplicar, según el tipo y el contexto del nodo seleccionado.
<u>xsl:attribute</u>	Crea un nodo de atributo y lo asocia a un elemento resultante.
<u>xsl:attribute-set</u>	Define un conjunto de atributos con nombre.
<u>xsl:call-template</u>	Invoca una plantilla por nombre.
<u>xsl:choose</u>	Proporciona múltiples pruebas condicionales junto con los elementos <code><xsl:otherwise></code> y <code><xsl:when></code> .
<u>xsl:comment</u>	Genera un comentario en el resultado.
<u>xsl:copy</u>	Copia el nodo actual del origen al resultado.
<u>xsl:copy-of</u>	Inserta subárboles y fragmentos del árbol de resultados en el árbol de resultados.
<u>xsl:decimal-format</u>	Declara un formato-digital, que controla la interpretación de un modelo de formato utilizado por la función <code>format-number</code> .
<u>xsl:element</u>	Crea un elemento con el nombre especificado en el resultado.
<u>xsl:fallback</u>	Llama al contenido de la plantilla que puede proporcionar un sustituto razonable al comportamiento del nuevo elemento cuando se encuentre.

RESUMEN ELEMENTOS XSLT (2)

<u>xsl:for-each</u>	Aplica una plantilla repetidamente, aplicándola a su vez en cada nodo de un conjunto.
<u>xsl:if</u>	Permite obtener fragmentos de plantillas condicionales simples.
<u>xsl:import</u>	Importa otro archivo XSLT.
<u>xsl:include</u>	Incluye otro archivo XSLT.
<u>xsl:key</u>	Declara una clave para utilizar con la función <code>key()</code> en expresiones de lenguaje de ruta XML (XPath).
<u>xsl:message</u>	Envía un mensaje de texto al búfer del mensaje o al cuadro de diálogo del mensaje.
<u>xsl:namespace-alias</u>	Sustituye el prefijo relacionado con un espacio de nombres dado por otro prefijo.
<u>xsl:number</u>	Inserta un número con formato en el árbol de resultados.
<u>xsl:otherwise</u>	Proporciona múltiples pruebas condicionales junto con los elementos <code><xsl:choose></code> y <code><xsl:when></code> .
<u>xsl:output</u>	Especifica las opciones que se deben utilizar a la hora de serializar el árbol de resultados.
<u>xsl:param</u>	Declara un parámetro con nombre que se puede utilizar dentro de un elemento <code><xsl:stylesheet></code> o un elemento <code><xsl:template></code> . Permite especificar un valor predeterminado.
<u>xsl:preserve-space</u>	Conserva los espacios en blanco en un documento.
<u>xsl:processing-instruction</u>	Genera una instrucción de proceso en el resultado.

RESUMEN ELEMENTOS XSLT (3)

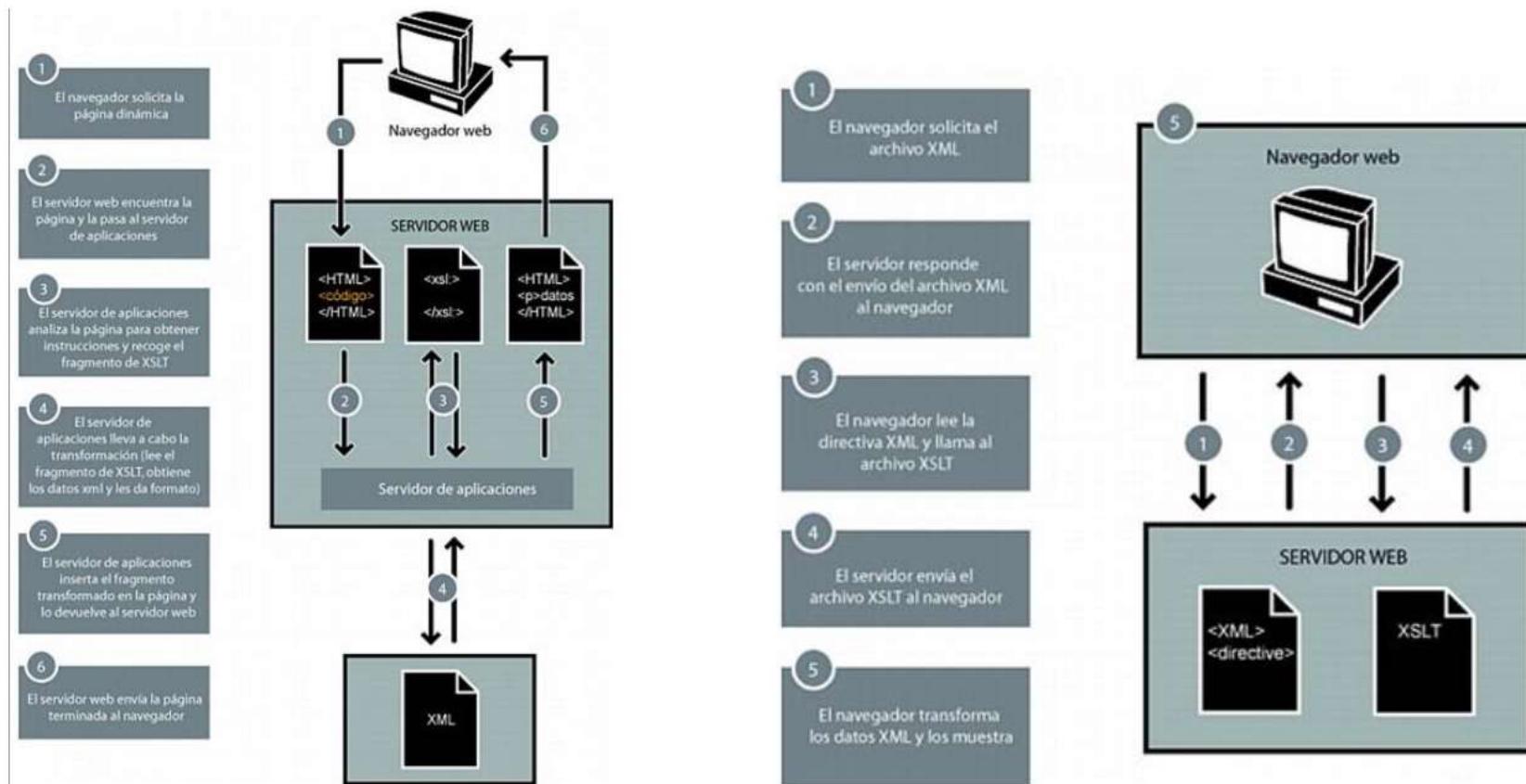
msxsl:script*	Define variables y funciones globales para extensiones de secuencias de comandos.
xsl:sort	Especifica los criterios de ordenación para las listas de nodos seleccionadas por <code><xsl:for-each></code> o <code><xsl:apply-templates></code> .
xsl:strip-space	Elimina espacios en blanco en un documento.
xsl:stylesheet	Especifica el elemento de documento en un archivo XSLT. El elemento de documento contiene otros elementos XSLT.
xsl:template	Define una plantilla reutilizable para generar los resultados deseados para los nodos de un tipo y contexto en particular.
xsl:text	Genera texto en el resultado.
xsl:transform	Lleva a cabo la misma función que <code><xsl:stylesheet></code> .
xsl:value-of	Inserta el valor del nodo seleccionado como texto.
xsl:variable	Especifica un valor enlazado de una expresión.
xsl:when	Proporciona múltiples pruebas condicionales junto con los elementos <code><xsl:choose></code> y <code><xsl:otherwise></code> .
xsl:with-param	Pasa un parámetro a una plantilla.

Resumen de funciones XPATH

- Existe una gran cantidad de funciones que se pueden usar en las expresiones XPATH
- Tipos:
 - Funciones de nodo → name(), node(), text()...
 - Funciones de posición → position(), last(), count(), ...
 - Funciones numéricas → number(), sum(), ...
 - Funciones booleanas → boolean(), not(), true(), false()
 - Funciones de cadena → string(), string-length(), contains(), ...

Uso de XSLT en Dreamwaver

- Dreamweaver proporciona métodos para crear páginas XSLT que admitan transformaciones XSL en el lado del servidor y en el lado del cliente



Crear una página XSL

- Se pueden crear páginas XSL o fragmentos (sin html) que permitan mostrar datos XML en páginas Web.

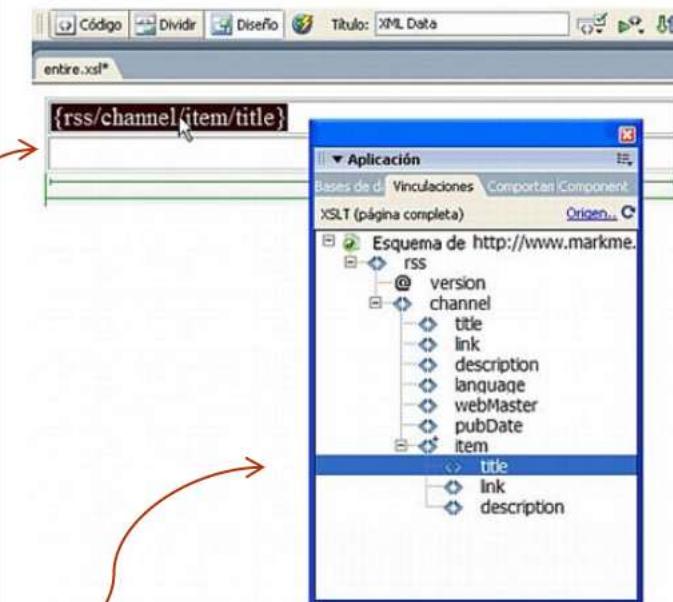
Para crear una página XSL:

1. Configure un sitio de Dreamweaver.
2. Seleccione Archivo > Nuevo.
3. En la ficha General del cuadro de diálogo Nuevo documento, seleccione Página básica en la columna Categoría y, después, realice una de las siguientes opciones:
 1. Seleccione XSLT (página completa) en la columna Página básica para crear una página XSLT completa.
 2. Seleccione XSLT (fragmento) en la columna Página básica para crear un fragmento de XSLT.
4. Haga clic en Crear. Aparece el cuadro de diálogo Buscar origen XML para adjuntar una fuente de datos XML.
5. Seleccione Adjuntar un archivo local en mi equipo o en la red de área local
6. Guarde la nueva página (Archivo > Guardar) con la extensión .xsl o .xslt (.xsl es la extensión predeterminada).

Visualización de datos XML en páginas XSL

Para mostrar los datos XML:

1. Abra una página XSL con una fuente de datos XML adjunta.
2. (Opcional) Seleccione Insertar > Tabla para añadir una tabla a la página.
3. En la mayoría de los casos, conviene utilizar el objeto XSLT Repetir región para mostrar elementos XML repetidos en una página. Si éste fuera el caso, puede resultar conveniente crear una tabla de una sola fila con una o varias columnas, o bien una tabla de dos filas si desea incluir un encabezado de tabla.
4. En el panel Vinculaciones, seleccione un elemento XML y arrástrelo hasta el lugar de la página en el que desea insertar los datos.
5. Obtenga una vista previa de su trabajo en un navegador (Archivo > Vista previa en el navegador)



XSLT Y CSS

- Podemos utilizar una hoja de estilos CSS en nuestra transformación del documento XML, para presentar su contenido.
- En nuestro documento xsl, cargaremos la hoja de CSS mediante:

```
.....  
<xsl:template match="/">  
  <html xmlns="http://www.w3.org/1999/xhtml">  
    <head>  
      <meta http-equiv="Content-Type" content="text/html; charset=iso-  
        8859-1"/>  
      <title>Untitled Document</title>  
      <link href="simple.css" rel="stylesheet" type="text/css" />  
    </head>  
    <body>  
      .....
```

Actividades

- Utilizando Dreamwaver crear un documento XSL para tiendecilla.xml que genere un documento HTML donde aparezca en una tabla todos los producto de la misma indicando el código, artículo, marca y modelo, ordenados por marca.
- Añadir una hoja de estilo CSS para que dicho documento HTML presente la mejor apariencia.

Conversión de páginas HTML en páginas XSLT

- Abra la página HTML que desea convertir.
- Seleccione Archivo > Convertir > XSLT 1.0.

Se abre una copia de la página en la ventana de documento con extensión xsl

- Una vez convertida podemos vincular un documento xml:
Para ello seleccionar Ventana/vinculaciones

Ya podemos añadir datos xml a nuestra página XSL

Actividad: Crear una página html, convertirla en xsl y mostrar datos de tiendecilla.xml en la misma mediante una tabla.

Actividad Final xml-xsl-css

- Crear un fichero XML que almacene datos de un diccionario bilingüe inglés – español.
- Debe almacenar los siguientes datos:
 - Palabra con atributo indicando el tipo (sustantivo, adjetivo, verbo, adverbio, etc)
 - Transcripción fonética
 - Traducción1
 - Traducción2
 - Imagen de referencia si es posible (nombre de un fichero .gif . El fichero debe estar almacenado en mismo directorio o carpeta que el fichero XML
 - Ejemplo en frase en inglés
- Debe almacenar al menos 15 palabras
- Crear un documento XSL llamado diccionario1.xsl que muestre una tabla con todos los datos incluso la imagen
- Crear un documento XSL llamado verbos.xsl que muestre una tabla con la palabra en inglés y su traducción que sea verbo
- Crear un documento XSL llamado sustantivos.xsl junto con una hoja de estilos sustativos.css que muestre una tabla con la palabra en inglés y su traducción que sea sustantivo