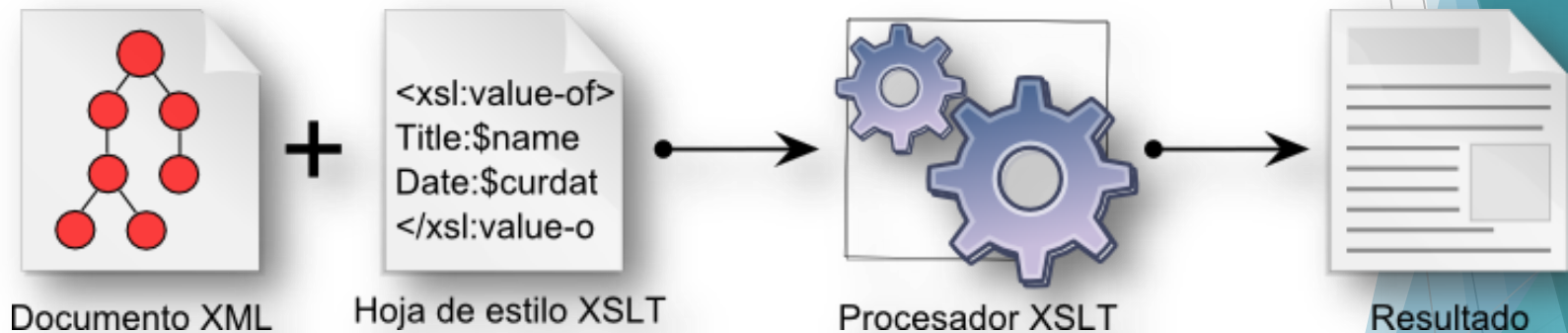


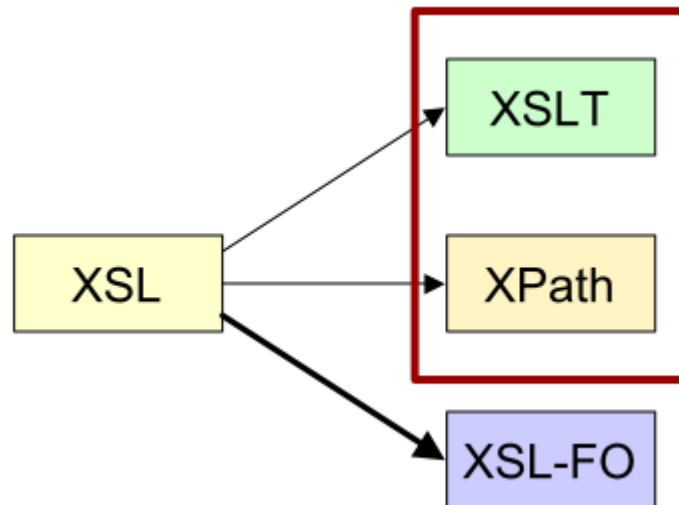
Unidad 5: Conversión y adaptación de documentos XML. XSLT



1. INTRODUCCIÓN

Los documentos XML son documentos de texto con etiquetas que contienen exclusivamente información sin entrar en detalles de formato. Esto implica la necesidad de algún medio para expresar la transformación de un documento XML para que una persona pueda utilizar directamente los datos para leer, imprimir, etc

XSL(eXtensible Stylesheet Language) es un especificación de W3C consistente en un estándar de hojas de estilo especialmente creado para la notación XML. En realidad es una combinación de los siguientes estándares:



XSLT

eXtensible Stylesheet Language
Transformations

3 INTRODUCCIÓN

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.

ejemplo1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="html/xsl" href="ejemplo1.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
    <company>RCA</company>
    <price>9.90</price>
    <year>1982</year>
  </cd>
</catalog>
```

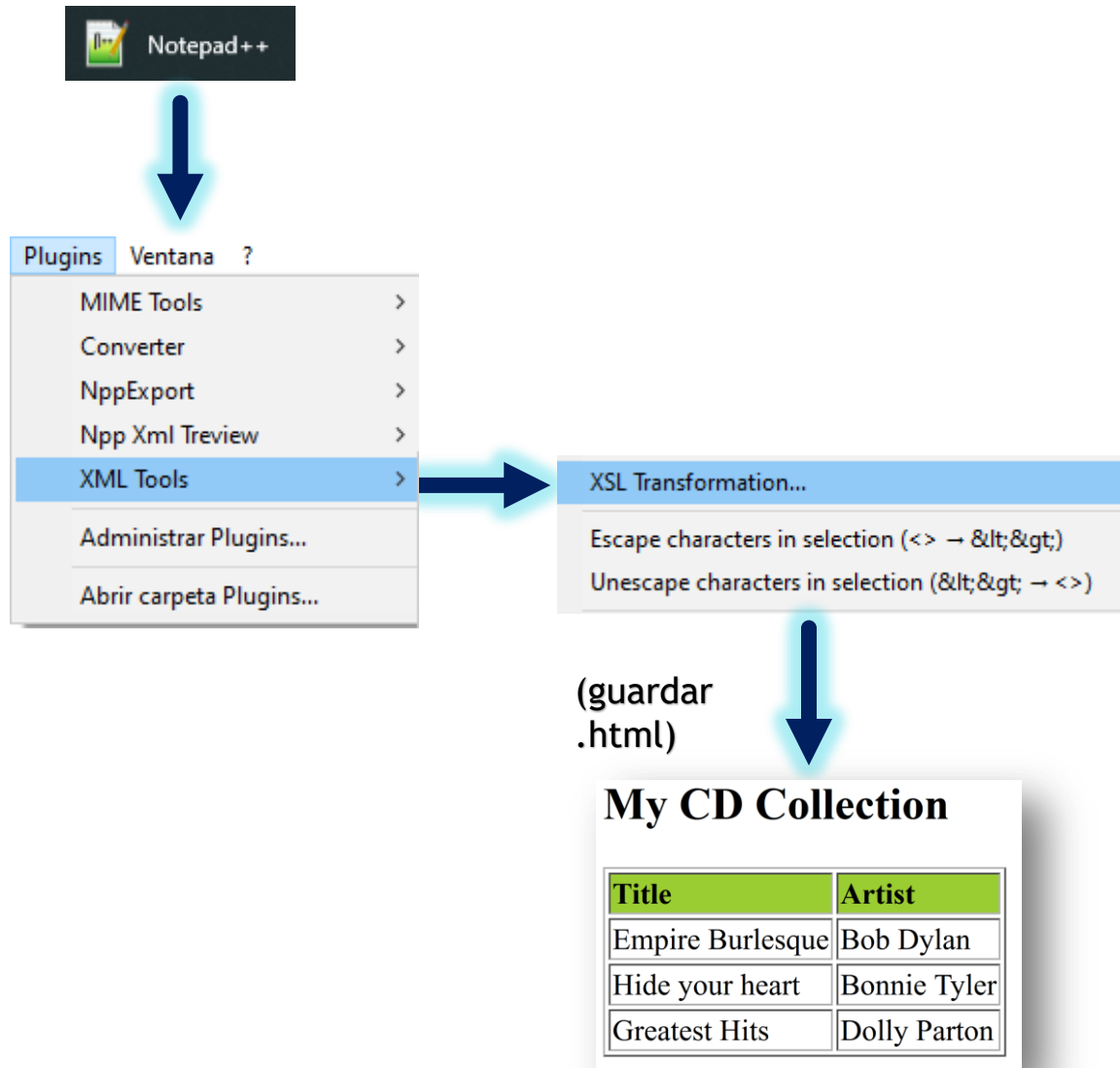
Tipo al que transformar y asociación con .xsl . Se puede transformar a txt, html, xml, etc.



ejemplo1.xsl

```
<?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>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th style="text-align:left">Title</th>
      <th style="text-align:left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
    </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

XSLT Transform



3.1 XSL

XSLT es un lenguaje declarativo. Por ello, las hojas de estilo XSLT no se escriben como una secuencia de instrucciones, sino como una colección de plantillas (**template rules**). Cada plantilla establece cómo se transforma un determinado elemento (definido mediante expresiones **XPath**). La transformación del documento se realiza de la siguiente manera:

1. El procesador analiza el documento y construye el árbol del documento.
 2. El procesador recorre el árbol del documento desde el nodo raíz.
 3. En **cada nodo recorrido**, el procesador aplica o no alguna plantilla:
 - Si a un nodo no se le puede aplicar ninguna plantilla, su contenido se incluye en el documento final (el texto del nodo, no el de los nodos descendientes). A continuación, el procesador recorre sus nodos hijos.
 - Si a un nodo se le puede aplicar una plantilla, se aplica la plantilla. La plantilla puede generar texto que se incluye en el documento final. En principio, el procesador no recorre sus nodos hijos, salvo que la plantilla indique al procesador que sí que deben recorrerse los nodos hijos.
- Cuando el procesador ha recorrido el árbol, termina la transformación.

3.2 Introducción elementos XSL

```
<?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' />
  <xsl:template match="/">
    <datos>
      <name>
        <xsl:value-of select="/pag/persona/nombre" />
      </name>
    </datos>
  </xsl:template>
</xsl:stylesheet>
```

La instrucción **<xsl:stylesheet>** es la etiqueta raíz de la hoja de estilo, sus atributos indican la versión y el espacio de nombres correspondiente. Dentro de la instrucción **<xsl:stylesheet>** se pueden encontrar los llamados elementos de alto nivel y las plantillas:

- El elemento de alto nivel **<xsl:output>** indica el tipo de salida producida. En nuestro ejemplo se generará un xml. No obligatorio.
- La instrucción **<xsl:template>** es una plantilla. El atributo match indica los elementos afectados por la plantilla y contiene una expresión **XPath**.
- El contenido de la instrucción define la transformación a aplicar (si la instrucción no contiene nada sustituirá el nodo por nada, es decir, eliminará el nodo, aunque conservará el texto contenido en el elemento).

Ejemplos de plantillas XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
  <libro>
    <titulo>La vida está en otra parte</titulo>
    <autor>Milan Kundera</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Pantaleón y las visitadoras</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1973"/>
  </libro>
  <libro>
    <titulo>Conversación en la catedral</titulo>
    <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
    <fechaPublicacion año="1969"/>
  </libro>
</biblioteca>
```

Ejemplos de plantillas XSLT

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

XSLT

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

</xsl:stylesheet>
```

Resultado

```
<?xml version="1.0" encoding="UTF-8"?>

  La vida está en otra parte
  Milan Kundera

  Pantaleón y las visitadoras
  Mario Vargas Llosa

  Conversación en la catedral
  Mario Vargas Llosa
```

- Si hay una **plantilla vacía**, el procesador no genera ningún resultado en el documento final ni recorre los nodos hijos.

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="autor">
  </xsl:template>

</xsl:stylesheet>
```

Resultado

```
<?xml version="1.0" encoding="UTF-8"?>

  La vida está en otra parte

  Pantaleón y las visitadoras

  Conversación en la catedral
```

Ejemplos de plantillas XSLT

- 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

```
<?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>
```

Resultado

```
<?xml version="1.0" encoding="UTF-8"?>
```

3.2 Introducción elementos XSL (II)

Cuando se aplica una plantilla a un nodo, en principio no se recorren los nodos descendientes. Para indicar que sí queremos recorrer los nodos descendientes y aplicarles las plantillas que les correspondan, hay que utilizar la instrucción `<xsl:apply-templates />` :

```
<?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:apply-templates />
  </xsl:template>

  <xsl:template match="elemento1">
  </xsl:template>

  <xsl:template match="elemento2">
  </xsl:template>

  <xsl:template match="elemento3">
  </xsl:template>

</xsl:stylesheet>
```

3.3 Uso de las plantillas (**template**)

Las plantillas van a determinar lo que mostrará en la transformación. Tenemos varios casos posibles. En los sucesivos ejemplos se utilizará el siguiente XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type='xml/xsl' href='ejemplo2.xsl'?>
<pag>
  <titulo>Eso es un título</titulo>
  <persona id="1">
    <nombre>Luis</nombre>
    <apellido>López</apellido>
  </persona>
  <persona id="2">
    <nombre>Teresa</nombre>
    <apellido>Trujillo</apellido>
  </persona>
  <persona id="3">
    <nombre>Almudena</nombre>
    <apellido>Pérez</apellido>
  </persona>
</pag>
```

3.3.1 Casos en el uso de las plantillas (**template**)

1. **Sin plantillas:** Se recorren todos los nodos y se muestra el texto

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

</xsl:stylesheet>
```

2. **Plantilla vacía:** No se procesa el contenido definido para esa plantilla

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

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

3. Si la plantilla **aplicase a /** y estuviese también vacía, no se mostraría nada:

```
<?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>
```

3.4 Selección en plantilla `<xsl:value-of>`

La instrucción `<xsl:value-of>` extrae el contenido del nodo seleccionado. Permite evaluar una expresión **Xpath**. El contenido del nodo actual viene dado por la expresión `"."` Solo tiene en cuenta los elementos seleccionados en **match**, y se aplicará tantas veces como la cantidad de elementos haya en ese nivel. Esto quiere decir que, si por ejemplo, en **match** seleccionamos `/pag`, solo se aplicará la plantilla una vez, mientras que si seleccionamos `persona` se aplicará la plantilla 3 veces, independientemente de lo que haya dentro. Ejemplos:

Ejemplo 1: Solo primer nodo encontrado

```
<?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:value-of select="pag/persona/nombre"/>
  </xsl:template>
</xsl:stylesheet>
```



XPATH
/pag/persona/nombre

De esta forma se forma la expresión XPATH `/pag/persona/nombre`. Devolverá así el primer nodo encontrado, en nuestro caso el nombre Luis.

3.4 Selección en plantilla `<xsl:value-of>` (II)

Ejemplo 2: Varios resultados

```
<?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:value-of select="pag/titulo" />
    <xsl:value-of select="pag/persona/nombre" />
    <xsl:value-of select="pag/persona/apellido" />
  </xsl:template>
</xsl:stylesheet>
```

Se pueden hacer varios select seguidos, observa que en la salida no se muestran espacios.

Ejemplo 3: Otra forma de usar select

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/pag/persona">
    <xsl:value-of select="nombre"/>
  </xsl:template>
</xsl:stylesheet>
```

En este caso vamos directamente a los nodos persona y para cada persona seleccionamos un nombre. Ahora sí se mostrarán todos los nombres (si está repetido nombre en persona solo la primera). La clave está en el atributo `match`.

3.4 Selección en plantilla <xsl:value-of> (III)

Ejemplo 4: Forma alternativa

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="persona">
    <xsl:value-of select="nombre"/>
  </xsl:template>
</xsl:stylesheet>
```

El atributo match permite seleccionar directamente el nodo sin la necesidad de indicar toda la ruta.

Ejemplo 5: Nodos no seleccionados

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/pag/persona">
    <xsl:value-of select="nombre" />
  </xsl:template>
  <xsl:template match="/pag/titulo">
    <xsl:value-of select="titulo" />
  </xsl:template>
</xsl:stylesheet>
```

Si un nodo no tiene plantilla mostrará su contenido y en caso de tener pero estar vacía, no lo muestra. Ya no se mostrará título

3.4 Selección en plantilla `<xsl:value-of>` (IV)

Ejemplo 6: Predicados XPATH

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/pag/persona[1]">
    <xsl:value-of select="nombre"/>
  </xsl:template>
</xsl:stylesheet>
```

Es posible aplicar todo lo que sabemos sobre XPATH, incluyendo los predicados. Observa que la salida de este ejemplo mostrará el nombre el primer nodo persona, el nombre y apellidos del segundo nodo persona y título. Esto es porque match únicamente hace referencia a la primera persona, de la que seleccionamos su nombre, y lo demás se mostrará igualmente si no tiene plantilla.

Ejemplo 7: Atributos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="persona">
    <xsl:value-of select="@id" />
  </xsl:template>
</xsl:stylesheet>
```

Para mostrar un atributo simplemente seleccionar el nodo y @ con el nombre.

3.4 Selección en plantilla **<xsl:value-of>** (V)

Ejemplo 8: Uso de match. Compara los resultados de estos dos xsl:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="persona">
    <xsl:value-of select="nombre" />
  </xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/pag">
    <xsl:value-of select="persona/nombre" />
  </xsl:template>
</xsl:stylesheet>
```

En el primer caso match coincide con tres resultados, por lo que se aplicará la plantilla a esos tres. En el segundo caso solo coincide con uno, por lo que se aplicará una única vez.

Trabajo clase 2. Selección de elementos

Ya sabemos la forma en la que se utiliza XPATH en XSL. Dado movies.xml del trabajo 1 y utilizando <template> y <value-of> genera mediante XSL otro fichero que muestre la siguiente información:

1. Los títulos de todas las películas.
2. Los títulos y el primer actor de todas las películas.
3. Todos los datos de todas las películas.
4. Todos los datos de la primera película.
5. Todos los datos de la última película.
6. Los comentarios de la penúltima película.
7. Películas con review igual a 5.
8. Películas con review igual a 5 y año igual a 1992.
9. Películas en la que ha actuado Nicolas Cage.
10. Productores de películas de 1992.
11. Un número con la cantidad de películas.
12. El atributo type de cada película.
13. El título de las películas de comedia.
14. El título en las que participe algún Coen

3.5 Añadir texto

Cuando se deseen añadir elementos fijos como por ejemplo otras etiquetas, texto, etc., simplemente se escribe en el xsl, transformamos y guardamos .html.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="pag">
    <html>
      <body>
        <h1><xsl:value-of select="titulo" /></h1>
        <table border="1">
          <tr bgcolor="#887788">
            <td colspan="1">¿Por qué solo aparece una persona al generar?</td>
          </tr>
          <tr>
            <td><xsl:value-of select="persona/nombre" /></td>
            <td><xsl:value-of select="persona/apellido" /></td>
          </tr>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

3.6 Selección en bucle `<xsl:for-each>`

Con la etiqueta `<value-of>` estamos limitados a un único resultado. Para evitar este inconveniente tenemos la etiqueta `<for-each>` que nos permitirá iterar sobre el nodo que elijamos:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="/pag">
    <xsl:for-each select="persona">
      <xsl:value-of select="nombre"/>
      <xsl:value-of select="apellido"/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

La clave está en iterar el nodo que tiene los subnodos que se quieren mostrar.

Es posible anidar varios bucles.

3.6 Selección en bucle **<xsl:for-each>** (II)

En este otro ejemplo se puede apreciar como en la etiqueta template ya no es necesario realizar el predicado, el cual se puede realizar perfectamente en el bucle, evitando que queden nodos sin plantilla:

```
<?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:for-each select="pag/persona[1]">
      <xsl:value-of select="nombre" />
      <xsl:value-of select="apellido" />
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```


3.7 Ordenación <xsl:sort>

Los elementos se procesan implícitamente en el orden de la selección, o bien en el orden indicado explícitamente por la directiva **sort**. Si lo que interesa es ordenarlos por el apellido, tras la etiqueta “for-each” incluimos:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="/pag">
    <xsl:for-each select="persona">
      <xsl:sort select="apellido" />
      <xsl:value-of select="nombre" />
      <xsl:value-of select="apellido" />
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Trabajo clase 3. For-each y generación de html

En el punto 3.5 generábamos un documento html, pero solo se mostraba a la primera persona. Con lo aprendido en el punto 3.6, haz que en la tabla se muestren a todas las personas.

Después, en el xml añade a las personas los elementos `<direccion>` y `<cod_postal>`. Amplia el html anterior para que en la tabla se muestren también estos campos. Realiza la ordenación por el código postal.

3.8 Condicional <xsl:if>

Los operadores lógicos se pueden utilizar para cambiar el patrón de búsqueda o filtro. on los siguientes:

- Operador de igualdad (=): "="
- Operador de desigualdad (≠): "!="
- Operador menor que (<): "<"
- Operador mayor que (>): ">"

```
<?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:for-each select="pag/persona">
      <xsl:if test="nombre='Luis'">
        <xsl:value-of select="nombre" />
        <xsl:value-of select="apellido" />
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

3.9 Condicional <xsl:choose>

La instrucción **choose** elige una acción entre varias alternativas. Cada **xsl:when** marca un caso. Solo se ejecuta las acciones del caso cuya condición se cumpla. En el caso por defecto se usa **xsl:otherwise** (sin atributos)

```
<?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:for-each select="pag/persona">
      <xsl:choose>
        <xsl:when test="@id=1"> ←
          Es id uno - <xsl:value-of select="nombre" />
        </xsl:when>
        <xsl:when test="@id=2"> ←
          Es id dos - <xsl:value-of select="nombre" />
        </xsl:when>
        <xsl:otherwise> ←
          No es ni uno ni dos - <xsl:value-of select="nombre" />
        </xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

3.10 Atributos especiales <xsl:attribute>

Mostrar un atributo de un xml es sencillo, únicamente es necesario utilizar la @ como se ha visto hasta ahora. Pero generar un atributo en la salida requiere de un paso adicional, el uso de la etiqueta <attribute> :

```
<?xml version="1.0" encoding="UTF-8"?>
<licencias>
  <licencia>
    <nombre>Creative Commons By - Share Alike</nombre>
    <imagen>cc-bysa-88x31.png</imagen>
  </licencia>
</licencias>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="licencia">
    <p><img>
      <xsl:attribute name="src">
        <xsl:value-of select="imagen" />
      </xsl:attribute>
    </img></p>
  </xsl:template>
</xsl:stylesheet>
```

El atributo será insertado en

3.11 Elemento `<xsl:apply-templates>`

Se utiliza para decirle al procesador XSLT que busque la plantilla adecuada para aplicar según el tipo y el contexto de cada nodo seleccionado. Ejemplos:

```
<?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 select="movies/movie" />
      </body>
    </html>
  </xsl:template>
  <xsl:template match="movie">
    <h1>
      TITULO:
      <xsl:value-of select="title" />
    </h1>
    <h1>YEAR: </h1>
  </xsl:template>
</xsl:stylesheet>
```

3.11 Elemento `<xsl:apply-templates>` (II)

Se pueden tener varias plantillas y usarlas cuando convenga:

```
<?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>
      <h1>PELICULAS</h1>
      <xsl:apply-templates select="movies/movie/title" />
      <h1>ACTORES:</h1>
      <xsl:apply-templates select="movies/movie/actor" />
    </body></html>
  </xsl:template>
  <xsl:template match="title">
    <h1>TITULO:</h1>
    <h4><xsl:value-of select="." /></h4>
  </xsl:template>
  <xsl:template match="actor">
    <h4><xsl:value-of select="." /></h4>
  </xsl:template>
</xsl:stylesheet>
```

3.11 Elemento `<xsl:apply-templates>` (III)

Se pueden concatenar entre sí:

```
<?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>
      <h1>PELICULAS</h1>
      <xsl:apply-templates select="movies/movie/title" />
      <h1>ACTORES:</h1>
      <xsl:apply-templates select="movies/movie/actor" />
    </body></html>
  </xsl:template>
  <xsl:template match="title">
    <h1>TITULO:</h1>
    <h4><xsl:value-of select="." /></h4>
  </xsl:template>
  <xsl:template match="actor">
    <h4><xsl:value-of select="." /></h4>
  </xsl:template>
</xsl:stylesheet>
```


3.11 Elemento **<xsl:apply-templates>** (IV)

Más ejemplos:

- <https://www.javatpoint.com/xslt-xsl-apply-template-element>
- <https://www.daniel.prado.name/Programacion-XSL-XSLT.asp?art=140>
- <https://www.mclibre.org/consultar/xml/ejercicios/xslt-1-soluciones.html>

Trabajo clase 4. Condicionales y atributos

Dado Movies.xml del primer trabajo, transfórmalo para que genere una tabla con los siguientes requisitos:

- Primera fila con los títulos en negrita y color de fondo azul.
- Campos Título, type, rating, director y todos los actores (en la misma celda).
- Si la película tiene rating R, la fila tendrá color de fondo rojo, si tiene PG-13 amarillo.
- El atributo year será un atributo en la etiqueta <tr> de cada fila. Por ejemplo, para la primera película: <tr year="1992"> El contenido </tr>