# LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

Bloque XML: XSLT

# XSLT

EXtensible Stylesheet Language Transformations

# XSLT



- Parte de XSL.
- Es XML
- Se usa para transformar
  - Un documento XML en otro documento XML,
  - O XML en otro documento que pueda ser reconocido por un navegador (HTML o XHTML).
  - O  XML a texto
- Con XSLT se pueden añadir o eliminar elementos y atributos.
- También ordenar, realizar test, tomar decisiones sobre qué elementos ocultar o mostrar, etc.
- Se suele decir que XSLT transforma un árbol XML fuente en un árbol XML resultado.

# XSLT: Ejemplos



- XML a XML
  ```
  $> xmlstarlet tr modifica.xsl catalog.xml
  ```
- XML a texto
  ```
  $> xmlstarlet tr genera-csv.xsl catalog.xml
  ```
- XML a HTML
  - ejemplo de ficheros RSS
    - Ver en navegador
      - Sin xsl asociado:
        ```
        $ firefox rss.xml
        ```
      - Con xsl asociado
        ```
        $ firefox rss_con_xsl.xml
        ```
    - En CLI
      ```
      $ xmlstarlet tr rsspretty.xsl rss.xml
      ```

**Error CORS...**

# La plantilla inicial



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

¡Es XML!

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

Especifica salida

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

Se aplica si encuentra expresión XPATH

```
    <xsl:text >RESULTADO</xsl:text>
    o texto sin etiquetas
  </xsl:template>
```

Genera nodo de texto en la salida, sin etiqueta también

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

# La plantilla inicial



```
<?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" encoding="utf-8" indent="yes"/>
 <xsl:template match="/">
    <!-- Comentario en XSLT -->
    <xsl:comment> Comentario credos desde XSLT </xsl:comment>

    <xsl:element name="elementoNuevo">
        <otroElemento> Contenido textual </otroElemento>
    </xsl:element>

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

Salida indentada

Genera comentario en la salida

Genera elementos de dos formas:
- xsl:element
- directamente

# Acceder a elemento o atributo



```
...
 <xsl:output method="xml" encoding="utf-8" indent="yes"/>

  <xsl:template match="/CATALOG">

    <coleccion>
      <propietario>
        <xsl:value-of select="OWNER"/> -
        <xsl:value-of select="OWNER/@country"/>
       </propietario>
      <localiza> <xsl:value-of select="URL/URL"/> </localiza>
    </coleccion>

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

Acceder valor elemento ( atributo con @)
- **<xsl: value-of select=""/>**
- XPATH en select

**localización** en cada momento:
**match="/CATALOG"**
**select="OWNER"**
**select="OWNER/@country**
**select="URL/URL**

# XSLT: value-of



- El elemento **<xsl:value-of>** se usa para extraer el valor del nodo seleccionado, y añadirlo al documento de salida resultante de la transformación.
- El atributo **select** contiene una expresión "Xpath".
  - Una expresión Xpath funciona de forma similar a navegar en un sistema de ficheros: la (/) selecciona subdirectorios, en este caso nodos hijos..
- Resultado del ejemplo anterior: sólo un par de líneas en la salida.
  - Queremos la información de cada CD
  - Vamos a mejorarlo con el elemento **<xsl:for-each>**
- Con **<xsl:for-each>** tendremos un bucle para iterar sobre los elementos XML y poder mostrar todos los elementos

# Bucle xsl:for-each select



```
...
 <xsl:output method="xml" encoding="utf-8" indent="yes"/>
 <xsl:template match="/CATALOG">
   <coleccion>
     <propietario>
       <xsl:value-of select="OWNER"/> -
       <xsl:value-of select="OWNER/@country"/>
     </propietario>
     <localiza> <xsl:value-of select="URL/URL"/> </localiza>
     <!--     Bucle para recorrer el catálogo      -->
     <xsl:for-each select="CD">
       <disco><xsl:value-of select="ARTIST"/>
              <xsl:text>: </xsl:text><xsl:value-of select="TITLE"/>
       </disco>
     </xsl:for-each>
   </coleccion>
 </xsl:template>
</xsl:stylesheet>
```

# XSLT: for- each



- El elemento <xsl:for-each> permite hacer un "bucle" en XSLT.
- Se usa para seleccionar cada uno de los elementos de un conjunto especificado de nodos del árbol XML.
- Observe con detenimiento
  - Atributo **match** del elemento **xsl:template**
  - Atributo **select** del elemento **xsl:for-each**
  - Atributo **select** del elemento **xsl:value-of**

# XSLT: sort



- Para ordenar, basta añadir el elemento **<xsl:sort>** dentro del elemento **<xsl:for-each>**

```
<xsl:for-each select="CATALOG/CD">
  <xsl:sort  select="ARTIST" />
  <disco><xsl:value-of select="ARTIST"/></disco>
```

- Otros atributos
  - data-type="**text**|number"
  - order="**ascending**|descending"
- Si desea ordenar por un segundo criterio se añade un segundo elemento sort

```
<xsl:for-each select="CATALOG/CD">
  <xsl:sort select="PRICE" data-type="number"/>
  <xsl:sort select="YEAR" />
```

# XSLT: **Filtrando la salida**



- Podemos filtrar los elementos que se recorren en el bucle xsl:for-each usando predicados XPATH en el atributo select
- Operadores válidos son:
  - `=` (equal)
  - `!=` (not equal)
  - `<` less than
  - `>` greater than

Entidades correspondientes a **<** y **>**.

- Ejemplo:

```
<xsl:for-each
      select="cd[artist='Bob Dylan']">
```

Observe el uso de las comillas simples y dobles..

# XSLT: Ejemplos filtros



- Filtrar discos de 1987
- Filtrar discos anteriores o posteriores a un año
- Filtrar discos de precio inferior o superior
- Etc..

# XSLT: Selectivas: <xsl:if>



- Sintaxis

```
<xsl:if  test="expresión">
    salida si la expresión es verdadera...
</xsl:if>
```

- Ejemplo

```
<xsl:for-each select="CD">
  <xsl:sort select="YEAR"/>
  <disco>
     <xsl:if test="PRICE&lt;8">¡OFERTA! </xsl:if>
     <xsl:value-of select="PRICE"/>
```

- No existe else … usar xsl:choose
- Se puede anidar con xsl:choose si es necesario

# XSLT: choose



- Es una selectiva múltiple

```
<xsl:choose>
  <xsl:when test="expression1">
    ... salida...
  </xsl:when>
  <xsl:when test="expression2">
    ... salida...
  </xsl:when>

    ...
  <xsl:otherwise>
    ... Si no se cumple ninguna
  </xsl:otherwise>
</xsl:choose>
```

# XSLT: choose



```
<disco>
   <xsl:choose>
     <xsl:when test="PRICE&lt;8">¡OFERTA!
     </xsl:when>
     <xsl:when test="PRICE&lt;10">¡Buen precio!

     </xsl:when>
     <xsl:otherwise>
        <xsl:text>¡Novedad! </xsl:text>
     </xsl:otherwise>
   </xsl:choose>
   <xsl:value-of select="PRICE"/>
...
```

Observe la salida que provoca esta línea en blanco tras el texto..

# Crear Atributo con valor tomado de instancia de datos: 3 formas

```
<xsl:element name="localiza">
      <xsl:attribute name="url">
           <xsl:value-of  select="location" />
      </xsl:attribute>
      <xsl:value-of  select="title" />
</xsl:element>
```

A

```
<localiza>
      <xsl:attribute name="url">
           <xsl:value-of  select="location" />
      </xsl:attribute>
      <xsl:value-of  select="title" />
</localiza>
```

B

```
<localiza url="{location}">
   <xsl:value-of select="title" />
</localiza>
```

C

# bucle que recorre elementos simples => acceder al nodo actual con "."



Se utiliza cuando el elemento a recorrer con el bucle es de tipo simple (no contenedor, como en ejemplo de los CDs.). En este ejemplo LINE

```
<CATALOG>
 ...
  <DESCRIPTION>
     <LINE>Creada en 2004.</LINE>
     <LINE>Se añaden CDs donados A Gar </LINE>
    ...
  </DESCRIPTION>
  <CD>
  ...
```

En este caso no se accede a un elemento hijo, sino al propio contenido del elemento. Para acceder al contenido se usa la expresión XPATH "."

```
<xsl:for-each select="DESCRIPTION/LINE">
  <p>
    <xsl:value-of select="."/>
  </p>
</xsl:for-each>
```

# bucles anidados



- Como en programación..
- Ejemplo: Cada CD tiene una BAND compuesta de músicos

```
<xsl:for-each select="CD">
   ...
   <banda>
      <xsl:for-each select="BAND/MUSICIAN">
         <xsl:value-of select="."/>,
      </xsl:for-each>
   </banda>
</xsl:for-each>
```

*Propuesta: ¿Cómo eliminamos la coma del último elemento?*

# Funciones auxiliares..



- concat()

```
<xsl:value-of
    select="concat(PRICE, '. Año: ', YEAR)" />
```

- position(), last()

```
<xsl:value-of select="position()"/>
<xsl:if test="position()=1">PRIMERO</xsl:if>
<xsl:if test="position()=last()">LAST</xsl:if>
```

- comprobar si un elemento o atributo existe:

```
<xsl:if test="price">
 <xsl:value-of select="concat('-',price,'-')"/>
</xsl:if>
```

# Transforma a CSV o HTML



- Si la salida es texto:

```
<xsl:output method="text"/>
```

- Generar saltos de línea, dos opciones
  - con salto de línea en texto

```
<xsl:text>
</xsl:text>
```

  - con entidades:

```
<xsl:text>&#10;</xsl:text>
```

- Si la salida es HTML:

```
<xsl:output method="html"/>
```

- Para enlazar instancia de datos con transformación a HTML:

```
<?xml-stylesheet type="text/xsl"
href="e1-html.xsl"?>
```

# CLI: xmlstartlet



- Transformar fichero datos.xml según transf.xsl.
  - Resultado se muestra en salida estándar
  - La opción -e muestra detalle errores transformación

```
$ xmlstartlet tr -e t.xsl d.xml
```

  - Si deseo almacenar en fichero el resultado:

```
$ xmlstartlet tr -e t.xsl d.xml > res.xml
$ more resultado.xml
   ...
```

- Nota: Generar un salto de línea:

```
<xsl:text> &#xa; </xsl:text>
```

# Ejercicios propuestos



- Modificar ejercicio de clase para mostrar músicos separados por comas, excepto último.
- Modificar ejercicio de clase para que atributo country de los artistas sea un atributo del elemento cancion.
- ...
- Ver enunciados ejercicios tipo examen en plataforma:
  - 01- Ejercicio proyecto
  - 03- Ejercicio bookmarks.
- Transformar a fichero de texto (por ejemplo .csv)
  - Ver ejercicio elecciones.xml

# Ejercicio completo 01



- Partimos de los ficheros `e1.xml` y el resultado de su transformación `e1-trasformado.xml`

**e1.xsl**: fichero utilizado para transformar el fichero e1.xml y obtener uno similar de modo que:

- el nuevo elemento raíz se llama memoria, y muestra en el elemento de nombre titular el título y su idioma en función del valor del atributo lang: Castellano, Francés, Inglés o Alemán).
- El elemento memoria lleva una atributo fecha con el valor de la fecha de publicación
- El elemento autores muestras los autores en el formato apellido, nombre, y separados por "y". Recuerde que el número de autores es variable,
- La bibliografía se mostrará como elementos "a", donde el atributo href será el destino del enlace.
- Los títulos de apartados aparecen como elementos h2, y los de sección como elementos h3
- Los títulos de apartado van precedidos de su identificador entre paréntesis.
- Los elementos párrafo se convierten a elementos "p"
- Solo se muestran los párrafos que no sean de la clase "revisar".

# Otra aproximación: apply-templates

NO VISTO CURSO 2020/21

# Otra aproximación: apply-templates



- Por defecto hay un template implícito que
  - para contenedores los recorre
  - para nodos de texto y atributos muestra el contenido
- `<xsl:apply-templates>` aplica la plantilla al elemento actual y a sus hijos
  - Si le añadimos el atributo `select`, procesa solo hijos que cumplan la condición del `select`.
    - Se puede utilizar para especificar el orden en el que se procesan los nodos hijos.

```
...
   <xsl:template match="/">
      <xsl:apply-templates/>
   </xsl:template>
</xsl:stylesheet>
```

# Ejemplo 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="/CATALOG">
    <miColeccion>
      <xsl:apply-templates select="CD"/>
    </miColeccion>
  </xsl:template>
  <xsl:template match="CD">
    <disco>
      <cancion>
        <xsl:value-of select="concat(TITLE, ', ', ARTIST)"/>
      </cancion>
      <xsl:apply-templates select="PRICE"/>
    </disco>
  </xsl:template>
  ...
</xsl:stylesheet>
```

con apply-templates se aplican

con template match se definen

```
<xsl:template match="PRICE">
  <precio>
    <xsl:if test=".&lt;10">¡Oferta!</xsl:if>
    <xsl:value-of select="."/>
  </precio>
</xsl:template>
```