UT6. XSL - Parte 2

LMSGI

Elementos básicos

Ejemplo **t06e14.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="/">
  <html>
   <xsl:apply-templates />
  </html>
 </xsl:template>

 <xsl:template match="libro">
 <p><img src="{imagen}"
          width="{imagen/@ancho}"
          height="{imagen/@alto}"></img></p>
 </xsl:template>

</xsl:stylesheet>
```

Para acceder a un atributo dentro de un elemento se utiliza **elemento/@atributo**

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
 <p><img src="1.png" width="301" height="200"/></p>
 <p><img src="2.png" width="300" height="200"/></p>
 <p><img src="3.png" width="300" height="200"/></p>
</html>
```

<xsl:text>

Ejemplo **t06e13.xsl**

```
<xsl:template match="/">
 <html>
  <xsl:text>&#10;</xsl:text>
  <xsl:text>Esto es un texto.</xsl:text>
  <xsl:text>&#10;</xsl:text>
  <h1>Autores</h1>
  <xsl:apply-templates />
 </html>
</xsl:template>
```

El elemento **<xsl:text>** inserta texto, aunque no es necesario ya que en las plantillas podemos escribir directamente el texto que queramos que se escriba en el árbol de resultado.

**
** → inserta un salto de línea.

```
<?xml version="1.0" encoding="UTF-8"?>
<html>Esto es un texto.

<h1>Autores</h1>
 <p>Milan Kundera</p>
 <p>Mario Vargas Llosa</p>
 <p>Mario Vargas Llosa</p>
</html>
```

<xsl:element> y <xsl:comment>

Ejemplo **t06e16a.xsl**

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

  <xsl:template match="/">
    <xsl:comment>Esto es un comentario!!</xsl:comment>
    <xsl:apply-templates />
  </xsl:template>

  <xsl:template match="libro">
    <libro>
      <xsl:element name="cantidad"><xsl:attribute name="stock">si</xsl:attribute>10</xsl:element>
    </libro>
  </xsl:template>

</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Esto es un comentario!!-->


<libro>
  <cantidad stock="si">10</cantidad>
</libro>


<libro>
  <cantidad stock="si">10</cantidad>
</libro>


<libro>
  <cantidad stock="si">10</cantidad>
</libro>
```

<xsl:copy-of>

Ejemplo **t06e16b.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="/">
  <xsl:apply-templates />
 </xsl:template>

 <xsl:template match="libro">
  <libro>
   <xsl:copy-of select="titulo" />
  </libro>
 </xsl:template>

</xsl:stylesheet>
```

El elemento **<xsl:copy-of>** toma un fragmento xml del origen y lo traspasa íntegramente al destino, sin necesidad de tener que crear los elementos.

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


<libro>
 <titulo>La vida está en otra parte</titulo>
</libro>


<libro>
 <titulo>Pantaleón y las visitadoras</titulo>
</libro>


<libro>
 <titulo>Conversación en la catedral</titulo>
</libro>
```

<xsl:copy />

Ejemplo **t06e16c.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="/">
  <xsl:apply-templates />
 </xsl:template>

 <xsl:template match="biblioteca">
    <xsl:copy />
 </xsl:template>

</xsl:stylesheet>
```

Este elemento, a diferencia del anterior, efectúa una **copia "hueca"** ya que sólo copia la etiqueta del elemento, mientras que los **atributos y nodos hijos son ignorados**.

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

<xsl:copy /> llenando elemento

Ejemplo **t06e16d.xsl**

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

  <xsl:output method="xml" />

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

  <xsl:template match="biblioteca">
    <xsl:copy>
      <todos_los_libros>
        <xsl:value-of select="." />
      </todos_los_libros>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

Si incorporamos **<xsl:value-of select=".">** se obtiene el contenido del nodo de contexto más el de todos los nodos descendientes

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

  La vida está en otra parte
  Milan Kundera

  1.png


  Pantaleón y las visitadoras
  Mario Vargas Llosa

  2.png


  Conversación en la catedral
  Mario Vargas Llosa

  3.png

</todos_los_libros>
</biblioteca>
```

Elementos de control

En XSLT se pueden usar **filtros y estructuras** que facilitan las operaciones de control del contenido de los documentos. Estas **operaciones de control son habituales en los lenguajes de programación tradicionales**, y se conocen como:

- **Iteraciones** (o repeticiones o bucles) → **for-each**
- **Selecciones** (o alternativas - si condición entonces ...) → **if test y chose**

La lógica que algorítmicamente corresponde a estas estructuras de control pueden ser complicada si no se ha practicado previamente algún lenguaje de programación, pero en este caso usaremos ejemplos con construcciones muy sencillas para que puedan ser entendidas sin ese requisito previo.

Además, se verá la forma de **ordenar elementos** de una iteración, mediante la instrucción de ordenación **sort**.

<xsl:for-each select=”expresión XPath” >

Ejemplo **t06e17a.xsl**

Este elemento se usa para **repetir la búsqueda de los nodos que coinciden con la expresión XPath** que se usa en el atributo **select** , de forma que sólo escribimos una vez el código que comprende la instrucción, pero **se aplica a todos los casos en que se cumpla la expresión**.

La sintaxis es: `<xsl:for-each select=”expresión XPath”>`

```
<xsl:template match="/biblioteca">
  <html>
  <body>
    <xsl:for-each select="libro">
      <xsl:value-of select="autor" />
    </xsl:for-each>
  </body>
  </html>
</xsl:template>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<html>
 <body>Milan KunderaMario Vargas LlosaMario Vargas Llosa</body>
</html>
```

<xsl:for-each select=”expresión XPath” >

Ejemplo **t06e17b.xsl**

```
<xsl:template match="/biblioteca">
 <html>
 <body>
  <xsl:text>&#10;</xsl:text>
  <xsl:for-each select="libro">
   <xsl:value-of select="autor" />
   <xsl:text>&#10;</xsl:text>
  </xsl:for-each>
 </body>
 </html>
</xsl:template>
```

```
<?xml version="1.0" encoding="UTF-8" sta
<html>
 <body>
Milan Kundera
Mario Vargas Llosa
Mario Vargas Llosa
</body>
</html>
```

<xsl:for-each select=”expresión XPath” >

Ejemplo **t06e17c.xsl**

```
<xsl:template match="/biblioteca">
 <html>
 <body>
 <xsl:for-each select="libro[fechaPublicacion/@anyo=1973]">
  <xsl:value-of select="autor" />
 </xsl:for-each>
 </body>
 </html>
</xsl:template>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<html>
 <body>Milan KunderaMario Vargas Llosa</body>
</html>
```

Como vemos, en esta ocasión hemos incluido una expresión **XPath** más compleja. Quiere decir que seleccionaremos **aquellos libros cuyo año de fecha de publicación es 1973**.

<xsl:for-each select=”expresión XPath” >

Ejemplo **t06e17d.xsl**

```
<xsl:template match="/">
 <html>
 <body>
 <xsl:apply-templates />
 </body>
 </html>
</xsl:template>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<html>
 <body>
 Milan Kundera
 Mario Vargas Llosa

  Conversación en la catedral
  Mario Vargas Llosa

  3.png

</body>
</html>
```

```
<xsl:template match="libro[fechaPublicacion/@anyo=1973]">
   <xsl:value-of select="autor" />
</xsl:template>
```

<xsl:for-sort >

Ejemplo **t06e18.xsl**

```
<xsl:template match="/biblioteca">
 <html>
 <body>
 <xsl:for-each select="libro[fechaPublicacion/@anyo=1973]">
  <xsl:sort select="titulo" order="descending"/>
  <xsl:value-of select="autor" />
 </xsl:for-each>
 </body>
 </html>
</xsl:template>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<html>
 <body>Mario Vargas LlosaMilan Kundera</body>
</html>
```

Este elemento se usa para **alterar el orden de presentación de los elementos**, siguiendo un criterio de clasificación indicado en el atributo select, y **se añade anidado dentro de un elemento <xsl:for-each....>, como primer hijo de ese elemento**. Su sintaxis completa es:

```
<xsl:sort select=”expresión XPath” lang=”lang-code” data-type=”text|number”
order=”ascending|descending” "/>
```

Selecciones: if test y chose

Disponemos también de elementos que nos permiten **decidir si una acción se realizará o no dependiendo de ciertos criterios**, es decir, **generar contenido condicionalmente**.

La sintaxis básica es:

```
<xsl:if test="expression Xpath">
.....
</xsl:if>
```

El atributo **test** es obligatorio y contiene, en **sintaxis XPath la condición** que se tiene que cumplir.

<xsl:if test=”expresión Xpath”>

Ejemplo **t06e19.xsl**

```
<xsl:template match="/biblioteca">
 <html>
 <body>
 <xsl:for-each select="libro">
  <xsl:if test="fechaPublicacion/@anyo=1973">
   <xsl:value-of select="autor" />
  </xsl:if>
  <xsl:if test="fechaPublicacion/@anyo!=1973">
   <xsl:value-of select="titulo" />
  </xsl:if>
 </xsl:for-each>
 </body>
 </html>
</xsl:template>
```

En este ejemplo, sólo se pintarán el autor y el título de los libros del año de publicación 1973.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<html>
  <body>Milan KunderaMario Vargas LlosaConversación en la catedral</body>
</html>
```

Selecciones: <xsl:chose>

Si queremos disponer de alguna alternativa en caso de que no se cumpla la condición, necesitaríamos utilizar la construcción chose, que además de poder elegir entre varias alternativas, nos deja planificar la acción a realizar si no se cumple ninguna de ellas.

La sintaxis básica es:

En esta estructura podemos tener tantas opciones when text=... como sean necesarias, y el funcionamiento será de tal forma que se irán comprobando secuencialmente sus valores. Si ninguna de ellas se cumple, y existe el elemento otherwise, que es optativo, entonces, esta sería la acción que se realizaría.

```
<xsl:choose>
        <xsl:when test="expresión booleana">
                .....
         </xsl:when>
        <xsl:when test="expresión booleana">
                .....
         </xsl:when>
                .....
        <xsl:otherwise>
                .....
         </xsl: otherwise>

</xsl:xhoose>
```

Selecciones: <xsl:chose>

Ejemplo **t06e20.xsl**

```
<xsl:template match="/biblioteca">
 <html>
 <body>
 <xsl:for-each select="libro">
  <xsl:choose>
   <xsl:when test="fechaPublicacion/@anyo=1973">
    <xsl:value-of select="autor" />
   </xsl:when>
   <xsl:otherwise>
    <xsl:value-of select="titulo" />
   </xsl:otherwise>
  </xsl:choose>
 </xsl:for-each>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

En este ejemplo, para cada libro con fecha de publicación en el año 1973, se escribe su autor, para el resto de libros se escribe su título.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<html>
 <body>Milan KunderaMario Vargas LlosaConversación en la catedral</body>
</html>
```