UNIDAD 5

Introducción

El principal objetivo de esta unidad es trabajar con ficheros XML base a partir del cual podremos obtener otros documentos XML que contengan partes del documento original, nuevos documentos HTML, añadirle o eliminarle etiquetas o atributos en el documento final, reorganizar sus elementos....

Para poder hacer todo esto emplearemos XPath (para poder acceder y referenciar los elementos del XML) y XSL (Extensible Stylesheet Language) que nos permitirá realizar las transformaciones XSLT (XSL Transformations).

XPath

- XPath es una sintaxis para definir partes de un documento XML
- XPath emplea expresiones para navegar en los documentos XML
- XPath contiene una librería de funciones estándar
- XPath es una pieza clave en XSL
- XPath es una recomendación del W3C

Los documentos XML son tratados como árboles de nodos. El nodo más alto del árbol es llamado el elemento root.

Sintaxis XPath

XPath emplea expresiones de trayectoria o camino para seleccionar nodos o conjuntos de nodos en un documento XML.

El nodo será seleccionado siguiendo la trayectoria o pasos indicados.

Exemplo de documento XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <bookstore>
 <book>
  <title lang="eng">Harry Potter</title>
  <price>29.99</price>
</book>
 <book>
  <title lang="eng">Learning XML</title>
  <price>39.95</price>
 </book>
 </bookstore>
```

Selección de Nodos

XPath emplea expresiones de ruta o camino para seleccionar nodos o conjuntos de nodos en un documento XML. Las rutas más usadas son:

Expresión	Descripción
/	Selecciona desde el nodo raíz.
//	Selecciona nodos en el documento desde el nodo actual que indica la selección.

.	Selecciona el nodo actual.
..	Selecciona el nodo padre del nodo actual.
@	Selecciona atributos.
text()	Selecciona el contenido del elemento (texto)

Ejemplos:

Expresión XPath	Resultado
/bookstore	Selecciona el elemento raíz bookstore.
/book	No devuelve nada por que book no es nodo raíz
/bookstore/book	Selecciona todos los elementos book que son hijos de bookstore
//book	Selecciona todos los elementos book no importa donde se encuentren en el documento.
/bookstore//book	Selecciona todos los elementos book que son hijos de bookstore, no importa donde estén debajo del elemento bookstore.
//@lang	Selecciona todos los atributos que tienen de nombre lang
//price/@*	No devuelve nada price no tiene atributos
//title/text()	Devuelve el contenido de title.

Predicados

Los predicados se usan para encontrar un nodo específico o un nodo que contiene un valor específico. Se ponen siempre entre corchetes

Ejemplos:

Expresión XPath	Resultado
/bookstore/book[1]	Selecciona el primer elemento book que es hijo del elemento bookstore.
/bookstore/book[last()]	Selecciona el último elemento book que es hijo del elemento bookstore.
/bookstore/book[last()-1]	Selecciona el elemento book anterior al último que es hijo del elemento bookstore.
/bookstore/book[position()<3]	Selecciona los dos primeros elementos book que son hijos del elemento bookstore.
//title[@lang]	Selecciona todos os elementos title que tienen un atributo llamado lang.

//title[@lang='eng']	Selecciona todos los elementos title que tienen un atributo llamado lang con un valor de 'eng'.
/bookstore/book[price>35.00]	Selecciona todos los elementos book del elemento bookstore que tienen un elemento price con un valor mayor que 35.00
/bookstore/book[price>35.00] /title	Selecciona todos los elementos title de los elementos book del elemento bookstore que tienen un elemento price con un valor mayor que 35.00

Selección de nodos desconocidos

Comodín	Descripción
*	Selecciona cualquier nodo.
@*	Selecciona cualquier atributo de los nodos.
node()	Selecciona cualquier nodo .

Ejemplos:

Expresión XPath	Resultado
/bookstore/*	Selecciona todos los elementos hijos del elemento bookstore
//*	Selecciona todos los elementos en el documento.
//title[@*]	Selecciona todos los elementos title que tengan algún atributo.

Ejes XPath

Un eje define un conjunto de nodos relativos al nodo actual.

Nombre del Eje	Resultado
ancestor	Selecciona todos los antecesores del nodo actual.
ancestor-or-self	Selecciona todos los antecesores del nodo actual y también el nodo actual.
attribute	Selecciona todos los atributos del nodo actual.
child	Selecciona todos los hijos del nodo actual.
descendant	Selecciona todos los nodos descendentes del nodo actual.
descendant-or-self	Selecciona todos los nodos descendientes del nodo actual e también el nodo actual.
following	Selecciona todo en el documento a partir de la etiqueta de cierre del nodo actual.

following-sibling	Selecciona todos los hermanos a partir del nodo actual.
parent	Selecciona el padre del nodo actual.
preceding	Selecciona todo en el documento que está antes de la etiqueta de comienzo del nodo actual.
preceding-sibling	Selecciona todos los hermanos antes del nodo actual.
self	Selecciona el nodo actual.

Expresión de localización XPath

Una ruta de localización puede ser absoluta o relativa.

Una ruta absoluta comienza con (/) y una ruta relativa no. En ambos casos la ruta de localización consiste en uno o más pasos, cada uno separado por (/):

```
Ruta de localización absoluta:
/paso/paso/...

Ruta de localización relativa:
paso/paso/...
```

La sintaxis para un paso de localización es:

axisname::nodetest[predicate]

Ejemplos

Ejemplo	Resultado
child::book	Selecciona todos los nodos book que son hijos del nodo actual.
attribute::lang	Selecciona el atributo lang del nodo actual.
child::*	Selecciona todos los hijos del nodo actual.
attribute::*	Selecciona todos los atributos del nodo actual.
child::text()	Selecciona todos los nodos de texto hijos del nodo actual.
child::node()	Selecciona todos los nodos hijos del nodo actual.
descendant::book	Selecciona todos los nodos book descendientes del nodo actual.
ancestor::book	Selecciona todos los nodos book antecesores del nodo actual.
ancestor-or-self::book	Selecciona todos los nodos book antecesores del nodo actual y también el nodo actual si es un nodo "book".

Operadores XPath

Una expresión XPath devuelve un conjunto de nodos, una cadena, un Boolean o un número.

A continuación tenemos una lista de operadores que se pueden emplear en las expresiones Xpath:

Operador	Descripción	Ejemplo	Valor devuelto
|	Calcula dos conjuntos de nodos	//book | //cd	Devuelve un conjunto de nodos con todos los elementos "book" y "cd".
+	Adición	6 + 4	10
-	Sustracción	6 - 4	2
*	Multiplicación	6 * 4	24
div	Division	8 div 4	2
=	Igualdad	precio=9.80	verdadero si precio es 9.80, falso en otro caso.
!=	Distinto	precio != 9.80	verdadero si precio es distinto de 9.80, falso si precio es igual a 9.80
<	Menor que	precio < 9.80	verdadero si precio es menor que 9.80, falso si precio es igual o mayor que 9.80
<=	Menor o igual	precio <= 9.80	verdadero se precio es menor o igual a 9.80, falso si precio es mayor que 9.80
>	Mayor que	precio > 9.80	verdadero si precio es mayor que 9.80, falso si es igual o menor que 9.80
>=	Mayor o igual	precio >= 9.80	verdadero si precio es mayor o igual a 9.80, falso si es menor que 9.80
or	o	precio = 9.80 or precio = 9.70	verdadero si precio es igual a 9.80 o igual a 9.70, falso en otro caso.
and	y	precio>9.00 and precio < 9.90	verdadero si precio es mayor que 9.00 y menor que 9.90
mod	resto de la división	5 mod 2	1

Funciones XPath

Existen multitud de funciones XPath que podemos emplear para hacer cálculos numéricos, extraer textos, funciones booleanas, etc.

Funciones Numéricas

Función	Descripción
number(arg)	Devuelve el valor numérico del argumento. El argumento puede ser un valor booleano, cadena o conjunto de nodos. Por ejemplo: number("100") -> Resultado: 100
abs(num)	Devuelve el valor absoluto do argumento. Ejemplo: abs(-3.14) -> Resultado: 3.14

ceiling(num)	Devuelve el valor integer inmediatamente superior al argumento. Ejemplo: ceiling(3.14) -> Resultado: 4
floor(num)	Devuelve el valor integer inmediatamente inferior al argumento. Ejemplo: floor(3.14) -> Resultado: 3
round(num)	Redondea el argumento al integer más próximo. Ejemplo: round(3.14) -> Resultado: 3

Funciones de Agregado

Función	Descripción
count((item,item,...))	Devuelve el número de nodos
avg((arg,arg,...))	Devuelve la media de los argumentos. Ejemplo: avg((1,2,3)) -> Resultado: 2
max((arg,arg,...))	Devuelve el argumento mayor. Ejemplo: max((1,2,3)) -> Resultado: 3; max(('b','t')) -> Resultado: 't'
min((arg,arg,...))	Devuelve el argumento menor. Ejemplo: min((1,2,3)) -> Resultado: 1; max(('b','t')) -> Resultado: 'b'
sum(arg,arg,...)	Devuelve la suma de los valores numéricos

Funciones de Cadena

Función	Descripción
concat(string s1, string s2, string s3, ...)	Devuelve un único string resultado de concatenar los parámetros
contains(string s1, string s2)	Devuelve true se s1 contén a s2
normalize-space(string s)	Devuelve s con ls espacios normalizados (sin espacios consecutivos, etc.)
starts-with(string s1, string s2)	Devuelve true si s1 empieza por s2, falso en caso contrario
string-length(string s)	Devuelve el número de carácteres de s

---

XSL

XSL son las siglas de EXtenxible Stylesheet Language, y es un lenguaje de hojas de estilos para los documentos XML.

XML (ejemplo)

Queremos transformar el siguiente documento XML ("lista.xml") en un archivo HTML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

hoja de estilos XSL

Ahora creamos una hoja XSL ("lista.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>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>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>
```

Para referenciar la hoja de estilos XSL al documento XML añadiremos al documento la siguiente línea:

```
<?xml-stylesheet type="text/xsl" href="lista.xsl"?> (2ª línea del XML)
```

ELEMENTOS

1- Elemento < xsl:template >

El elemento < xsl:template > es usado para construir modelos.

El atributo match se usa para asociar el modelo con el elemento XML. El atributo match puede

ser usado para definir un modelo para el documento XML.

El valor del atributo match es una expresión XPath (por ejemplo match="/" define el documento completo).

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

2- Elemento <xsl:value-of>

El elemento <xsl:value-of> se emplea para extraer los valores de un nodo determinado del archivo XML, y enviar esos valores a la salida transformados.

Ejemplo: <xsl:value-of select="title"/>

3- Elemento <xsl:for-each>

El elemento <xsl:for-each> nos permite hacer bucles en el XSLT. Este elemento puede ser usado para seleccionar cada uno de los elementos dentro de un conjunto especificado.

Ejemplo: <xsl:for-each select="catalog/cd">

- **Filtrando a saída**

Podemos filtrar la salida desde el archivo XML añadiendo un criterio o atributo select en el elemento <xsl:for-each>.

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

Operadores disponibles en los filtros:

- = igual
- != distinto
- < menor que
- > mayor que

4- Elemento <xsl:sort>

El elemento <xsl:sort> se emplea para ordenar la salida.

Simplemente tendremos que añadir el elemento **<xsl:sort>** dentro de cada elemento <xsl:for-each> en el archivo XSL:

Ejemplo:
```
<xsl:for-each select="catalog/cd">
    <xsl:sort select="artist"/>
```

5- Elemento < xsl:if >

El elemento < xsl:if > se emplea para crear una condición a evaluar en el contenido XML.

Sintaxis:

```
<xsl:if test="expresion">
  ... salida cuando la expresiónes verdadera ...
</xsl:if>
```

Para añadir una condición, tendremos que añadir el elemento <xsl:if> dentro del elemento <xsl:for-each> en el archivo XSL.

Ejemplo:

```
<xsl:for-each select="catalog/cd">
 <xsl:if test="price &gt; 10">
   <tr>
     <td><xsl:value-of select="title"/></td>
     <td><xsl:value-of select="artist"/></td>
   </tr>
 </xsl:if>
```

6- Elemento <xsl:choose>

El elemento **<xsl:choose>** se emplea junto con **<xsl:when>** y **<xsl:otherwise>** para expresar condiciones múltiples.

Sintaxis:

```
<xsl:choose>
 <xsl:when test="expression">
 ... some output ...
 </xsl:when>
 <xsl:otherwise>
 ... some output ....
</xsl:otherwise>
 </xsl:choose>
```

Ejemplo:

```
<xsl:choose>

<xsl:when test="price &gt; 10">
<td bgcolor="#ff00ff">
<xsl:value-of select="artist"/></td>
</xsl:when>
<xsl:otherwise>
<td><xsl:value-of select="artist"/></td>
</xsl:otherwise>
</xsl:choose>
```

7- **Elemento <xsl:apply-templates>**

El elemento <xsl:apply-templates> aplica un modelo al elemento actual o a los nodos elementos hijos del actual.

Si añadimos un atributo select al elemento <xsl:apply-templates> procesará solamente el elemento hijo que coincida con el valor del atributo. Podemos usar un atributo select para especificar la orden en la cual serán procesados los nodos hijos.

Ejemplo:

```
<?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>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>

<xsl:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="artist">
  Artist: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

</xsl:stylesheet>
```