TEMA 8 (segunda parte)
Conversión y adaptación de documentos XML

Andrés López Martinez

Departamento de Informática – IES San Juan Bosco

Master formación del profesorado

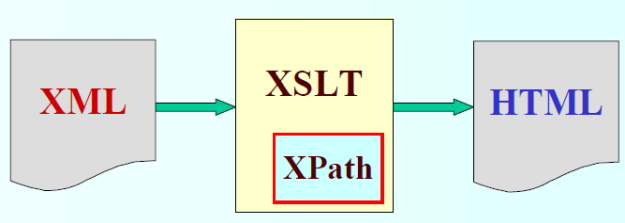
Tema 8

Conversión y adaptación de documentos XML

- Todo el procesamiento realizado con un fichero XML está basado en la posibilidad de direccionar o **acceder a cada una de las partes que lo componen**, de modo que podamos tratar cada uno de los elementos de forma diferenciada.
- La forma de seleccionar información dentro de él es mediante el uso de XPath, que es la abreviación de lo que se conoce como XML Path Language. Con XPath podremos seleccionar y hacer referencia a texto, elementos, atributos y cualquier otra información contenida dentro de un fichero XML.
- XPath sirve para decir cómo debe procesar una hoja de estilo el contenido de una página XML, pero también para poder poner enlaces o cargar en un navegador zonas determinadas de una página XML, en vez de toda la página.
- XPath en sí es un lenguaje sofisticado y complejo, de tipo declarativo, pero distinto de los lenguajes procedurales que solemos usar (C,C++, Basic, Java...).

Introducción a XPath

- XPath es una especificación del W3C (aprobada el mismo día que XSLT).
- Define cómo acceder a partes de un documento XML.
- Se basa en relaciones de “parentesco” entre nodos.
- Su estilo de **notación es similar a las rutas de los ficheros**, pero se refiere a nodos en un documento XML:
  - Ejemplo: /fecha/dia
- XPath se usa en XSLT, pero también en XSL-FO, XPointer, XLink, y otros.



- En XSLT, XPath se utiliza en los valores de atributos (atributos tales como **match** o **select**, empleados en las etiquetas **xsl:value-of** y **xsl:template** respectivamente)

Modelo de datos de Xpath (1)

- Un documento XML es procesado por un analizador (o *parser*) construyendo un árbol de nodos.
  - Este árbol comienza con un elemento raíz, que se diversifica a lo largo de los elementos que cuelgan de él y acaba en nodos hoja, que contienen sólo texto, comentarios, intrucciones de proceso o incluso que están vacíos y sólo tienen atributos.
- El árbol de nodos:
  - Comienza en el nodo raíz
  - Acaba en los nodos hoja
- XPath selecciona partes del documento XML basándose en la estructura en árbol.

Modelo de datos de Xpath (2)

```
/
|
+---libro
       |
       +---titulo
       |      |
       |      +---(texto)Dos por tres calles
       |
       +---autor
       |      |
       |      +---(texto)Josefa Santos
       |
       +---capitulo [num=1]
       |       |
       |       +---(texto)La primera calle
       |       |
       |       +---parrafo
       |       |      |
       |       |      +---(texto)Era una sombría noche ...
       |       +---parrafo
       |              |
       |              +---(texto)Ella, cual inocente mariposa...
       |
       +---capitulo [num=2]
```

```
<libro>
    <titulo>Dos por tres calles</titulo>
    <autor>Josefa Santos</autor>
    <capitulo num="1">
        La primera calle
        <parrafo>
            Era una sombria noche del mes de agosto...
        </parrafo>
        <parrafo destacar="si">
            Ella, inocente cual
            <enlace href="http://www.enlace.es">mariposa</enlace>
            que surca el cielo en busca de libaciones...
        </parrafo>
    </capitulo>
    <capitulo num="2" public="si">
```

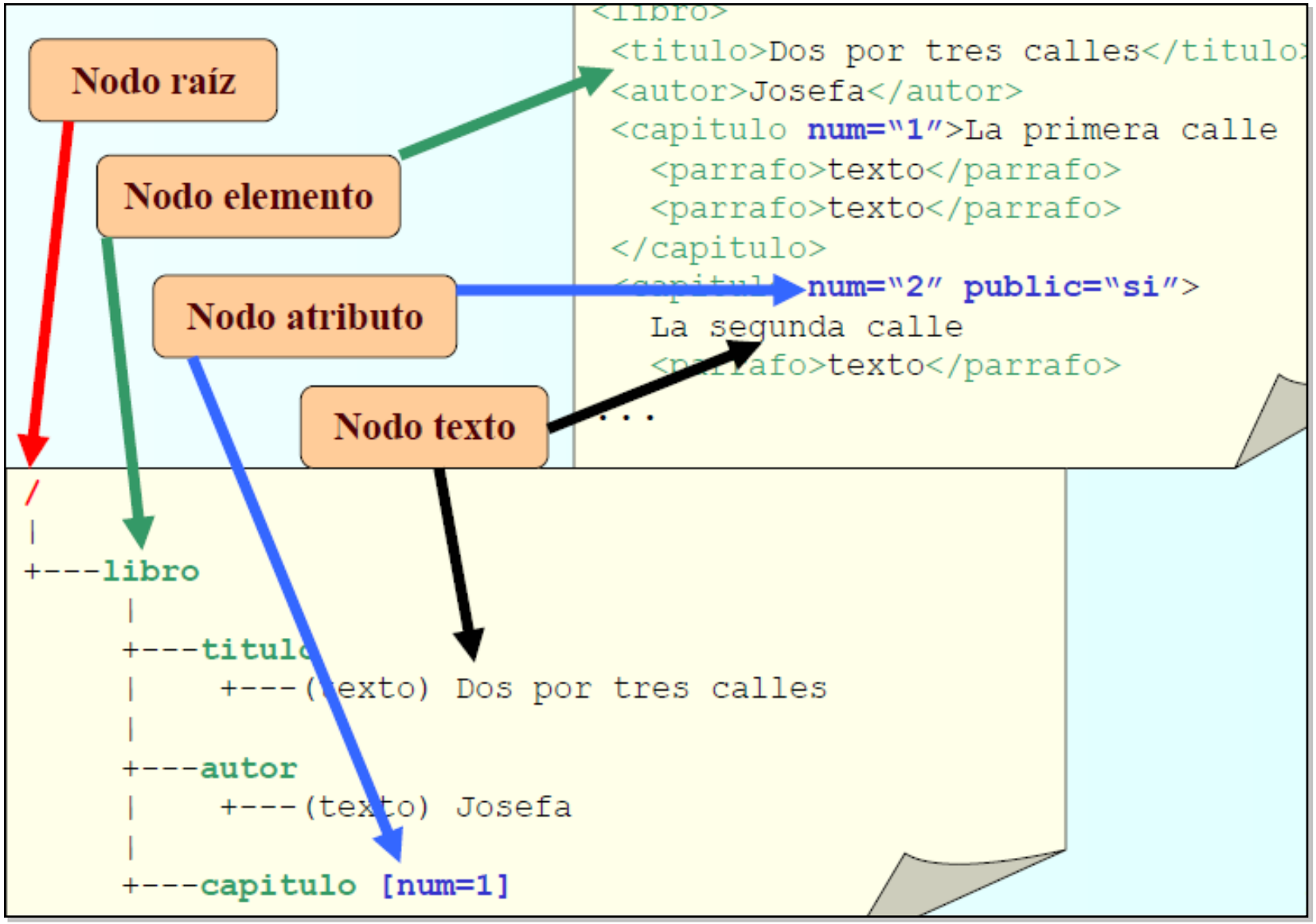
Tipos de nodos (1)

- **Nodo Raíz**
  - Se identifica por /. No se debe confundir el nodo raíz con el elemento raíz del documento.
    - El documento XML de nuestro ejemplo tiene por elemento raíz a libro, éste será el primer nodo que cuelgue del nodo raíz del árbol, el cual es: /.
- **Nodo Elemento**
  - Cualquier elemento de un documento XML se convierte en un nodo elemento dentro del árbol.
  - Cada elemento tiene su nodo padre. El nodo padre de cualquier elemento es, a su vez, un elemento, excepto el elemento raíz, cuyo padre es el nodo raíz.
  - Los nodos elemento tienen a su vez hijos, que puede ser nodos de cualquier tipo excepto raiz.

Tipos de nodos (2)

- **Nodos Texto**
  - referenecia a todos los caracteres del documento que no está marcados con alguna etiqueta.
- **Nodo Atributo**
  - Más que hijos del nodo elemento que los contiene son como etiquetas añadidas a dicho nodo.
  - Cada nodo atributo consta de un nombre, un valor (que es siempre una cadena).
- Nodos comentario, y de Instrucciones de proceso.
  - Al contenido de estos nodos se puede acceder con la propiedad string-value.

Tipos de nodo (3)



Expresiones XPath (1)

- Una "instrucción" en lenguaje XPath se denomina **expresión**.
- Hay que considerar una expresión XPath como un “predicado”, que devuelve todo lo que encaja con dicho predicado.
- Lo que devuelve es procesado por la regla XSL.
- Las expresiones XPath se usan sobre todo en los atributos match, select (y test).

Expresiones XPath (2)

- Una expresión XPath arroja (tras ser evaluada) una expresión que puede ser de 4 tipos posibles: **conjunto de nodos** (node-set), **booleano**, **número**, ó **cadena**.
- Tokens válidos en una expresión XPath
  - Paréntesis y similares: ( ) { } [ ]
  - Elemento actual . y elemento padre ..
  - Atributo @, * y separador ::
  - La coma ,
  - El nombre de un elemento
  - Tipo de nodo (comment, text, processing instruction, node)
  - Operadores: and, or, mod, div, *, /, //, |, +, -, =, !=, <, <=, >, >=
  - Nombres de función
  - Nombre de eje (axis): ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self
  - Literales, entre comillas dobles o simples (se pueden anidar alternadas)
  - Números
  - Referencias a variables ($nombreVariable)
- No centraremos en los que veamos en los ejemplos y ejercicios siguientes.

Location Paths

- Un ***location path*** es la más importante de los tipos de expresiones que se pueden especificar en notación XPath.
- La sintaxis de un location path es simliar a la usada a la hora de describir los directorios que forman una unidad de disco en Unix.

Location Paths. Sintaxis (1).

- Una ruta de directorio de Linux indica la ruta a un archivo particular:
  - Ejemplo: /home/juan/documentos
    - Referencia a un único directorio llamado *documentos* que cuelga de */home/juan*
- Location Paths se corresponde con la idea intuitiva de “ruta de directorio”, pero un location path siempre devuelve un *node-set*:
  - **XPath indica la ruta hasta varios nodos, basándose en la estructura del documento XML**
  - Ejemplo: **/libro/capitulo/parrafo**
    - Referencia a **todos** los elementos *parrafo* que cuelguen de cualquier *capitulo* del *libro*

```
<libro>
    <titulo>Dos por tres calles</titulo>
    <autor>Josefa Santos</autor>
    <capitulo num="1">
        La primera calle
        <parrafo>
            Era una sombria noche del mes de
        </parrafo>
        <parrafo destacar="si">
            Ella, inocente cual
            <enlace href="http://www.enlace.e
            que surca el cielo en busca de li
        </parrafo>
    </capitulo>
    <capitulo num="2" public="si">
    La segunda calle
        <parrafo>
            Era una obscura noche del mes de
        </parrafo>
        <parrafo>
            Ella, inocente cual
            <enlace href="http://www.abejilla
            que surca el viento en busca del
        </parrafo>
    </capitulo>
```

Location Paths. Sintaxis. (2)

- Si se indica el camino completo, la búsqueda comienza en el nodo raíz.
- Si se indica un camino completo, se entiende que el path comienza en el nodo que en cada momento se está procesando.
- Ejemplo:
  - /libro/capitulo/parrafo
    - Al leer ***/*** se selecciona el nodo raíz como nodo contexto.
    - Al leer ***libro*** se selecc. los elem libro que cuelgan del contexto (***/***)
    - Al leer ***capitulo*** se selecc los elem capitulo que cuelgan del contexto (en ese momento es ***libro***)
    - Al leer **parrafo** se selecc los elem parrafo que cuelgan del contexto (en ese momento es **capitulo**)
- http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~XPath_as_filesystem_addressing

Location Paths. Nodo Contexto (1)

- Un location path siempre tiene un nodo contexto
  - Es similar al concepto de directorio actual:
    - ls ./juan/documentos
- En XPath, si la expresión comienza por / estamos dando un path absoluto, partiendo del nodo raíz.
- Si no comienza por /, estamos dando un camino relativo desde el nodo actual (nodo contexto).

Location Paths. Nodo Contexto (2)

- Nodo actual (**current node**)
  - Es un nodo que está seleccionado cuando se va a evaluar una expresión XPath
  - Constituye el punto de partida al evaluar la expresión
- Nodo **contexto** (context node)
  - Para evaluar una expresión, se van evaluando subexpresion parciales
  - Cada vez que se evalúa una subexpresión se obtiene un nuevo conjunto de nodos (node-set) que es el nuevo contexto para evaluar la siguiente subexpresión
- Tamaño del contexto (context size)
  - El número de nodos que se están evaluando en un momento dado. Luego el Nodo contexto puede ser en si un conjunto de nodos, para los que se evaluaria una expresion empezando por uno de ellos, que sera el Current node en el momento de evaluar la expresion.
- *Estos conceptos pueden parecer oscuros, pero se vera, si no aparece antes, una evidencia de su importancia cuando hagamos el ejercicio con el fichero Horario5.xls*

Location Paths. Nodo Contexto (3)

- **Ejemplo con /libro/capitulo/parrafo**
- El analizador comienza por leer **/**, lo cual le dice que debe seleccionar el nodo raíz, independientemente del nodo contexto que en ese momento exista. En el momento en que el evaluador de XPath localiza el nodo raíz, éste pasa a ser el nodo contexto de dicha expresión.
- El analizador leería ahora **libro**, lo cual le dice que seleccione TODOS los elementos que cuelgan del nodo contexto (que ahora mismo es el nodo raíz) y que se llamen libro. En este caso solo hay uno... porque solo puede haber un elemento raíz. El nodo contxto pasa a ser libro.
- A continuación el analizador leería **capitulo**, entonces selecciona TODOS los elementos que cuelgan del nodo contexto (ahora es el nodo libro).
  - En un disco sería imposible que hubiera dos directorios con el mismo nombre colgando de un mismo directorio padre.
  - En estos momentos hay dos elementos que encajan con el patrón /libro/capitulo.
- El analizador continua leyendo **parrafo**. Con lo que selecciona TODOS los elementos parrafo que cuelgan del nodo contexto...¡¡pero NO hay un nodo contexto, sino DOS!!. El evaluador de expresiones lo va a recorrer uno por uno haciendo que, mientras evalúa un determinado nodo, ése sea el nodo contexto de ese momento.

Location Paths.

Un location path consta de:

- Eje (axis). Es la relación entre el nodo de contexto y el Location Path
  - El eje a veces está implícito (no se pone, y entonces se sobreentendera “es hijo de”).
  - Para nosotros esto sera lo común.
- Prueba de nodo (node test). Es el “nombre de directorio”
- Predicado (predicate). Expresión XPath entre corchetes.
  - El predicado es opcional

eje::pruebanodo[predicado]

Location Paths: Prueba de nodo (1)

eje::**pruebanodo**[predicado]

- La forma más simple es escribir simplemente el **nombre del nodo** (su etiqueta)
- * que simboliza cualquier nombre
- Ejemplos:
  - /libro/capitulo/parrafo
    - Encaja con cualquier nodo “parrafo” que sea hijo de un nodo “capitulo” que sea hijo de un nodo “libro” que será el nodo raíz
  - /libro/*
    - * simboliza cualquier nombre: encaja con cualquier nodo que sea hijo del nodo “libro” que será el hijo del nodo raiz.
  - capitulo/*
    - Encaja con cualquier nodo que sea hijo de un nodo “capitulo” que sea hijo del nodo de contexto
- IMPORTANTE: // indica “que sea hijo de cualquiera”
  - Los atributos se especifican con el prefijo @. Su acceso ser vera más adelante.
- A continuacion veremos mas ejemplos.

Location Paths: Prueba de nodo (2)

/AAA

Select the root element AAA

```
<AAA>
    <BBB/>
    <CCC/>
    <BBB/>
    <BBB/>
    <DDD>
        <BBB/>
    </DDD>
    <CCC/>
</AAA>
```

Location Paths: Prueba de nodo (3)

/AAA/CCC

Select all elements CCC which are children of the root element AAA

```
<AAA>
    <BBB/>
    <CCC/>
    <BBB/>
    <BBB/>
    <DDD>
        <BBB/>
    </DDD>
    <CCC/>
</AAA>
```

Location Paths: Prueba de nodo (4)

/AAA/DDD/BBB

Select all elements BBB which are children of DDD which are children of the root element AAA

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC/>
</AAA>
```

Location Paths: Prueba de nodo (5)

//BBB

Select all elements BBB

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
      </DDD>
   </CCC>
</AAA>
```

Location Paths: Prueba de nodo (6)

//DDD/BBB

Select all elements BBB which are children of DDD

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
      </DDD>
   </CCC>
</AAA>
```

Location Paths: Prueba de nodo (7)

/AAA/CCC/DDD/*

Select all elements enclosed by elements /AAA/CCC/DDD

```
<AAA>
   <XXX>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </XXX>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </CCC>
   <CCC>
      <BBB>
         <BBB>
            <BBB/>
         </BBB>
      </BBB>
   </CCC>
</AAA>
```

Location Paths: Prueba de nodo (8)

/*/*/BBB

Select all elements BBB which have 3 ancestors

```
<AAA>
   <XXX>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </XXX>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </CCC>
   <CCC>
      <BBB>
         <BBB>
            <BBB/>
         </BBB>
      </BBB>
   </CCC>
</AAA>
```

Location Paths: Prueba de nodo (9)

```
//*
```

Select all elements

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

¿Qué conjunto de nodos se seleccionaría si solo hubiera puesto ***/****?

Location path: Prueba de nodo(10)

- Evaluacion de expresiones Xpath on line:
  - http://www.whitebeam.org/library/guide/TechNotes/xpathtestbed.rhtm
  - http://www.zvon.org/comp/tests/r/test-xlab.html#XPath_1~Expressions
- A continuacion veremos ejemplos de Pruebas de nodo y **los probaremos en nuestro navegador**.
- **Pero antes de nada ¿Como probar expresiones fácilmente?**

Cómo probar expresiones (1)

Libro.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="evalua_xpath.xsl"?>
<libro>
    <titulo>Dos por tres calles</titulo>
    <autor>Josefa Santos</autor>
    <capitulo num="1">
        La primera calle
        <parrafo>
            Era una sombria noche del mes de agosto...
        </parrafo>
        <parrafo destacar="si">
            Ella, inocente cual
            <enlace href="http://www.enlace.es">mariposa</enlace>
            que surca el cielo en busca de libaciones...
        </parrafo>
    </capitulo>
    <capitulo num="2" public="si">
    La segunda calle
        <parrafo>
            Era una obscura noche del mes de septiembre...
        </parrafo>
        <parrafo>
            Ella, inocente cual
            <enlace href="http://www.abejilla.es">abejilla</enlace>
            que surca el viento en busca del nectar de las flores...
        </parrafo>
    </capitulo>°
    <apendice num="a" public="si">
        La tercera calle
        <parrafo>
            Era una densa noche del mes de diciembre...
        </parrafo>
        <parrafo>
            Ella, candida cual
            <enlace href="http://www.pajarillo.es">abejilla</enlace>
            que surca el espacio en busca de bichejos para comer...
        </parrafo>
    </apendice>
</libro>
```

1. Para realiar los ejemplos y ejercicios es suficiente con disponer de un navegador Web que sea capaz de mostrar documentos XML a los que, a su vez, se aplica un fichero XSL, esto es, un fichero que realiza transformaciones.
2. Descarga libro.xml de los contenidos del tema: http://dis.um.es/~lopezquesada/documentos/IES_1011/LMSGI/curso/material.html#ut8
3. *Libro.xml* incluye una línea que referencia al fichero XSL…

Cómo probar expresiones (2)

Libro.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="evalua_xpath.xsl"?>
<libro>
```

Evalua_xpath.xsl

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

  <xsl:template match="libro">
  <HTML>
    <HEAD>
      <TITLE>Ejemplos en XPath</TITLE>
    </HEAD>
    <BODY>
      <H1>Resultados:</H1>
      <PRE>
        <xsl:apply-templates
             select="/libro/capitulo/text()"/>
      </PRE>
    </BODY>
  </HTML>
  </xsl:template>

</xsl:stylesheet>
```

3. *Libro.xml* incluye una línea que indica qué fichero XSL que va a utilizarse para seleccionar nodos y generar la información con la que realimente se desea trabajar (y que nos permitirá práticar con evaluaciones de expresiones XPath)

4. Descarga del mismo sitio que antes el fichero *evalua_xpath.xsl*

5. A continuación, lo único que tienes que hacer es sustituir /libro/capitulo por la expresión que quieras evaluar y volver a guardar el fichero.

Ojo: no necesita acabar con /text()

6. Carga libro.xml con el navegador, y prueba.

Location Path: Prueba nodo (11)

*****

- Devuelve todos los nodos de tipo principal (es decir, elemento, atributo o espacio de nombres), pero no nodos de texto, comentarios y de instrucciones de proceso.
- Ejemplo: Seleccionar todos los nodos principales descendientes de los **parrafo**:
  - //parrafo/*

Location Path: Prueba nodo (12)

node()

- El predicado node() devuelve todos los nodos de todos los tipos.

- Ejemplo: Seleccionar todos los nodos descendientes de los **parrafo**:
  - //parrafo/node()

Location Path: Prueba nodo

text()

- Selecciona cualquier nodo de tipo texto (recuerda el ejemplo del fichero evalua_xpath.xls).
- Ejemplo:
  - Seleccionar el texto de todos los nodos parrafo:
    - //parrafo/text()
  - Seleccionar TODO el texto que cuelga de todos los nodos parrafo:
    - //parrafo//text()
  - ojo: ¿ves la diferencia con el anterior?

- Ejercicio: Dado el fichero **horario.xml**, generar una hoja HTML que muestre los días que aparecen (simplemente el número de día)
  - Realizar el ejercicio sobre el fichero dado: **horario1.xsl**

    (en adelante se especificara el nombre del fichero sobre el que hacer el ejercicio, tras el enunciado, evidentemente.).

Location Path: Predicado (1)

eje::pruebanodo[**predicado**]

- un predicado permite restringir el conjunto de nodos seleccionados a aquellos que cumplen cierta condición.
  - Dicha condición es una expresión XPath y se expecifica entre corchetes.
  - E una Expresión booleana: un nodo del conjunto seleccionado la cumplirá, o no la cumplirá.
- Dada la prueba de nodo, y dado el eje (supongamos “es hijo de”), del conjunto de nodos resultante quedan sólo los que cumplan el predicado
- En el predicado pueden intervenir **funciones Xpath,** las veremos mas adelante.
- Ver los ejemplos de: http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Further_conditions_inside_%5B%5D

Location Path: Predicado (2)

- Ejemplos para poder **probarlos con nuestro ficheros libro.xml y evalua_xpath.xsl**:
  - Ejemplo: Seleccionar todos los **capitulo** que tengan un **parrafo** que tenga algún elemento con atributo **href**:
    - //capitulo[parrafo/*[@href]]
  - Serán muchas las ocasiones en que deseemos que los nodos a seleccionar no cumplan una sino varias condiciones. En estos casos, los predicados se pueden suceder uno a otro haciendo el efecto de la operación AND. Como en el siguiente ejemplo.
  - Ejemplo: Seleccionar todos los **capitulo** que tengan un **parrafo** que tenga algún elemento con atributo **href** y que ellos mismos (los **capitulo**) tengan el atributo **public** a valor **si**:
    - //capitulo [parrafo/*[@href]] [@public='si']

Location Path: Predicado (3)

- también se puede hacer uso del operador and encerrando entre paréntesis los distintos predicados logicos. Ejemplo similar al anterior:
  - //capitulo[(parrafo/*[@href]) and (@public='si')]
  - También se puede hacer uso de la operación **OR** en vez de and.
- Existe otro tipo de operacion **or** que utiliza la barra vertical: **|** separando no dos predicados, sino dos expresiones XPath.
- Ejemplo de |: Seleccionar todos los **capitulo** que tengan un **parrafo** que tenga algún elemento con atributo **href** o todos los **apendice**:
  - //capitulo[parrafo/*[@href]] **|** //apendice
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Combining_XPaths_with_%7C

Location Path: Predicado (4)

- Por último, también podemos especificar con **not** la negación de alguna de las negaciones del predicado.
- Ejemplo: Seleccionar todos los **capitulo** que no tengan el atributo **public: //**capitulo[not(@public)]
- **Predicados con funciones de cardinalidad**
  - Existen funciones que nos van a servir para restringir los nodos basándose en la posición del elemento devuelto.
    - **position**(), → Ejemplo: //capitulo[position()=2] ó
      //capitulo[2]
    - **last**() →Ejemplo: //capitulo[last()] ó
      //capitulo[not(position()=last())] ó
      //capitulo[last()-1]
    - **id**(). →Ejemplo: id("capitulo_1")/parrafo

Location Path: Predicado (5)

- Ejercicio: crear una hoja para el horario en la que sólo salgan las tareasdespués del miércoles (día 3 en adelante)
  - horario2.xsl

Más funciones Xpath (1)

- Hay una gran variedad de **funciones** que podemos usar en el predicado:
  - **boolean**(): convierte a booleano. Aplicada a un conjunto de nodos, devuelve true si no es vacío. not(), true()
  - **count**(): Devuelve el número de nodos en un conjunto de nodos.
    - **Ver** http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Nodes_counting
  - **name**(): Devuelve el nombre de un nodo (su etiqueta). local-name(), namespace-uri()
    - **Ver** http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Playing_with_names_of_selected_elements
  - Biblioteca de strings:
    - **normalize-space**(), **string**(), **concat**(), **stringlength**()
    - **Ver** http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Length_of_string
  - **sum**()
  - Recuerda también las de **cardinalidad**: id(), last(), position()
- Ejercicio: Escribir una hoja que muestre en HTML todos los nodos de un documento, como listas no numeradas, indicando el número de orden de cada nodo y el número de hijos que contiene (cada elemento irá, además, numerado)
  - – horario3.xsl

Más funciones Xpath (2)

//*[count(BBB)=2]

Select elements which have two children BBB

```
<AAA>
   <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </CCC>
   <DDD>
      <BBB/>
      <BBB/>
   </DDD>
   <EEE>
      <CCC/>
      <DDD/>
   </EEE>
</AAA>
```

//*[count(*)=2]

Select elements which have 2 children

```
<AAA>
   <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </CCC>
   <DDD>
      <BBB/>
      <BBB/>
   </DDD>
   <EEE>
      <CCC/>
      <DDD/>
   </EEE>
</AAA>
```

//*[count(*)=3]

Select elements which have 3 children

```
<AAA>
   <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </CCC>
   <DDD>
      <BBB/>
      <BBB/>
   </DDD>
   <EEE>
      <CCC/>
      <DDD/>
   </EEE>
</AAA>
```

Más funciones Xpath (3)

//*[name()='BBB']

Select all elements with name BBB, equivalent with //BBB

```
<AAA>
   <BCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </BCC>
   <DDB>
      <BBB/>
      <BBB/>
   </DDB>
   <BEC>
      <CCC/>
      <DBD/>
   </BEC>
</AAA>
```

//*[starts-with(name(),'B')]

Select all elements name of which starts with letter B

```
<AAA>
   <BCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </BCC>
   <DDB>
      <BBB/>
      <BBB/>
   </DDB>
   <BEC>
      <CCC/>
      <DBD/>
   </BEC>
</AAA>
```

//*[contains(name(),'C')]

Select all elements name of which contain letter C

```
<AAA>
   <BCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </BCC>
   <DDB>
      <BBB/>
      <BBB/>
   </DDB>
   <BEC>
      <CCC/>
      <DBD/>
   </BEC>
</AAA>
```

Más funciones Xpath (4)

//*[string-length(name()) = 3]

Select elements with three-letter name

```
<AAA>
   <Q/>
   <SSSS/>
   <BB/>
   <CCC/>
   <DDDDDDDD/>
   <EEEE/>
</AAA>
```

//*[string-length(name()) < 3]

Select elements name of which has one or two characters

```
<AAA>
   <Q/>
   <SSSS/>
   <BB/>
   <CCC/>
   <DDDDDDDD/>
   <EEEE/>
</AAA>
```

//*[string-length(name()) > 3]

Select elements with name longer than three characters

```
<AAA>
   <Q/>
   <SSSS/>
   <BB/>
   <CCC/>
   <DDDDDDDD/>
   <EEEE/>
</AAA>
```

Localization Paths. Ejes (1)

eje::pruebanodo[predicado]

- El eje denota la relación de un Localization Paths con su nodo de contexto
- Hay una serie de ejes posibles: ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self.
- Equivale a “que es un”, pero sus argumentos se leen de derecha a izquierda
- **child está implícito** y casi nunca se pone.
  - Pero para el nodo raíz, está implícito self (self denota al nodo de contexto)
- Ejemplos (*no recreables con nuestro ejemplo libro.xml*):
  - /universidad/euitio
    - Equivale de manera implícita a /self::universidad/child::euitio
  - /universidad/euitio/following-sibling::*
    - Todos los nodos que son “hermanos después de” euitio (en el orden del documento) que es hijo de universidad

Localization Paths. Ejes (2)

Child

- Es el hacha utilizada por defecto. Se corresponde con la barra, / (aunque tiene una forma más larga que es: /child::).
- Ejemplo: Seleccionar todos los **titulo** de un **libro**:
  - /libro/titulo
- Sugerencia: Seleccionar el autor del libro.
- Sugerencia: Seleccionar todos los párrafos del libro.

Localization Paths. Ejes (3)

Attribute

- Se corresponde con el signo de la arroba, @ (o en su forma larga que es: attribute::). Mediante este operador podemos seleccionar aquellos nodos atributos que deseemos, indicando el nombre del atributo en cuestión.
- Ejemplo: Seleccionar el atributo **num** que posean los elementos **capitulo**
  - /libro/capitulo/@num
    - (Para ver este ejemplo tendrás que quitar /text() del fichero evalua_xpath.xsl)
- Ejemplo: Seleccionar todos los elementos hijo de los **capitulo** que posean el atributo **public** (sin importar el valor asignado al mismo):
  - /libro/capitulo[@public]/*
- Ejemplo: Seleccionar todos los elementos hijo de **parrafo** cuyo atributo **destacar** sea igual a "si".
  - /libro/titulo/parrafo[@destacar="si"]

Attribute

//@id

Select all attributes @id

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

//BBB[@id]

Select BBB elements which have attribute id

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

//BBB[@*]

Select BBB elements which have any attribute

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

//BBB[not(@*)]

Select BBB elements without an attribute

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

//BBB[@name]

Select BBB elements which have attribute name

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

Localization Paths. Ejes (4)

- **Recuerda** que se puede acceder a un elemento atributo gracias al eje attribute::
  - Contiene todos los nodos atributo del nodo contexto
  - Una abreviatura de esto es la arroba @
  - Más ejemplos:
    - Ver http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Attributes
    - Ver http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Attribute_values
- Ejercicio: Generar una versión del horario que para cada día muestra la lista de tareas (sus nombres) y su prioridad, y también la hora de inicio y fin.
  - horario4.xsl

Localization Paths. Ejes (6)

- No haremos ejercicios con los ejes, pero si os propongo ver más ejemplos.
- Ejes posibles: ancestor, ancestor-or-self, attribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self. Ejemplos:
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Child_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Descendant_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Parent_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Ancestor_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Following-sibling_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Preceding-sibling_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Following_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Preceding_axis
  - http://www.zvon.org/comp/r/tut-XPath_1.html#Pages~Descendant-or-self_axis
- Para prácticar: http://www.zvon.org/comp/tests/r/test-xlab.html#XPath_1~Axes

Localization Paths. Ejes (5)

Descendant

- Se especifica poniendo una doble barra: // (en su forma larga: descendant::).
- Sirve para seleccionar TODOS los nodos que descendiendan del conjunto de nodos contexto. Es decir, no solo los hijos de los nodos contexto, sino también los hijos de los hijos, y los hijos de estos, etc.
- Ejemplo: Seleccionar todos los **parrafo** de un **libro**:
  - /libro//parrafo
- Ejemplo: Seleccionar todos los descendientes de **parrafo** que tienen un atributo **href**.
  - //parrafo//*[@href]
- Ejemplo: Mostrar el valor del atributo **href** del caso anterior:.
  - //parrafo//*[@href]/@href

Localization Paths. Ejes (7)

Self

- Se especifica mediante el punto (.).
- Es muy útil pues sirve para seleccionar el nodo contexto.
- Para seleccionar todos los parrafo descendientes del nodo contexto no podemos escribir //parrafo, dado que seleccionaría todos los descendientes del nodo raíz. Por ello, la forma correcta es: .//parrafo

Parent

- Al igual que en los sistemas de ficheros, se utilizan los dos puntos para identificarlo: ..
- El comportamiento de este eje es un poco extraño al principio dado que realiza un paso hacia atrás en el árbol de nodos.
- Ejemplo: Seleccionar todos los nodos que tienen algún hijo de tipo **parrafo**:
  - //parrafo/..
- Ejemplo: Seleccionar todos los nodos **capitulo** que tienen algún hijo de tipo **parrafo**:
  - //parrafo/../../capitulo
  - O bien: //capitulo/parrafo/..

Localization Paths. Ejes (8)

Ancestor

- De todas los ejes que podemos usar, esta es la única que no tiene ninguna forma de abreviación. Siempre aparece como ancestor::
- Ancestor es a parent lo que descendant es a child. Es decir, devuelve todos los elementos de los cuales el nodo contexto es descendiente.
- Ejemplo: Seleccionar todos los elementos que tienen entre sus descendientes algún **parrafo**
  - //parrafo/ancestor::*

Acceso a elementos de otro documento XML

- Se puede acceder a datos de otro fichero XML:
  - Mediante la función **document**()
- Ejercicio: usando el fichero “literales.xml” generar una versión del horario (sobre horario4.xsl) que en vez de “Día 1” muestre “Lunes” y así sucesivamente
  - horario5.xsl.
  - NOTA: habra que usar
    concat(),
    current() y
    normalize-space())

```
<?xml version="1.0" encoding="windows-1252"?>
<literales>
    <dia1>Lunes</dia1>
    <dia2>Martes</dia2>
    <dia3>Miércoles</dia3>
    <dia4>Jueves</dia4>
    <dia5>Viernes</dia5>
    <dia6>Sábado</dia6>
    <dia7>Domingo</dia7>
</literales>
```

Ayuda para Horario5.xsl

- En vez de obtener el número de dia tal y como lo haciamos en horario4.xsl:
  - **<P>Día <xsl:value-of select="numdia"/></P>**

  tenemos que obtener su equivalente en literal (así, en vez de devolver "**Día 1**", devolveremos "**Lunes**", y asi sucesivamente).
- Los literales estan en literal.xml, luego la expresión que buscamos tendra este aspecto:

  <xsl:value-of **select="document('literales.xml') …** />
- Los literales estan dentro del nodo literales:

  select="document('literales.xml')/**literales/***
- Pero mediante un predicado hay que seleccionar el texto del nodo literal cuyo nombre corresponda al numero de dia que se esta procesando:

  select="document('literales.xml')/literales/***[name()= …]**"/>
- Los nombres de los nodos literales son como "diaX", por lo que habra que implementar la condicion de igualdad entre una cadena del tipo "diaX", siendo X el valor de numdia, y el nombre del nodo hijo de literal diaX:

  [name()=**concat('dia', normalize-space(current()/numdia))**]
- ¿Por que "**current()/numdia**" y "./numdia"?

  Cando se usan funciones anidades, debemos referirnos al nodo contexto con current(), en vez de con .

```
<?xml version="1.0" encoding="windows-1252"?>
<literales>
    <dia1>Lunes</dia1>
    <dia2>Martes</dia2>
    <dia3>Miércoles</dia3>
    <dia4>Jueves</dia4>
    <dia5>Viernes</dia5>
    <dia6>Sábado</dia6>
    <dia7>Domingo</dia7>
</literales>
```

Resumiendo…

- Con XPath podemos:
  - Seleccionar los nodos para la aplicación de templates.
  - Obtener valores (bastante elaborados)
  - La selección de nodos puede basarse en similitud de nombres, en el eje y/o en ciertas condiciones (predicado)

Instrucciones XSL (1)

- No son elementos de nivel superior; son las instrucciones contenidas dentro de los templates.
- Indican cómo realizar el procesamiento
  - xsl:value-of es un caso simple.
- Otras instrucciones permiten realizar tratamientos condicionales,iteraciones, construcción de elementos en el árbol resultado, etc:
  - Xsl:sort
  - Xsl:if
  - Xsl:choose, xsl:when, xsl:otherwise
  - Xsl:for-each

Instrucciones XSL (2)
Ordenar con xsl:sort

xsl:sort

- Se especifica dentro de xsl:apply-templates o xsl:for-each
- ¿Podría haber sido un atributo?
- Su atributo es select
- Indica cómo se establece el orden

•Ejercicio: hacer que el horario salga en orden

–horario6.xsl

EJEMPLO:

```
<xsl:apply-templates select="producto">
  <xsl:sort select="articulo" type="text"
            order="ascending"/>
  <xsl:sort select="precio" type="number"
            order="ascending"/>
</xsl:apply-templates>
```

Instrucciones XSL (3)
Condicional: xsl:if

xsl:if

- Atributo: **test**
  - = equal
  - != not equal
  - < less than
  - > greater than
- El valor del atributo es una **expresión booleana**
- Las instrucciones que contiene se ejecutan sólo si la condición se cumple

```
<xsl:if test='condición'>
…
</xsl:if>
```

- Ejercicio: hacer que en el horario no salga la “Prioridad” si realmente el elemento no tiene tal atributo
  - horario7.xsl
  - Ayuda: Donde debe aparecer la prioridad ahora, no siempre debe ser así:

```
<xsl:if test="¿?¿?">
    Prioridad: <xsl:value-of select="./@prioridad"/>
</xsl:if>
```

Instrucciones XSL (4)
Condicional: xsl:choose

xsl:choose

- Contiene elementos xsl:when
  - Atributo: test (similar al de xsl:if)
  - Son los diferentes “casos” de una sentencia CASE
  - Caso por defecto: xsl:otherwise (sin atributos)

```
<xsl:choose>
      <xsl:when test="condición">
      …
      </xsl:when>
      <xsl:when test="condición">
      …
      </xsl:when>
      <xsl:otherwise>
      …
      </xsl:otherwise>
</xsl:choose>
```

Instrucciones XSL (5)
Iteración: xsl:for-each

xsl:for-each

- Atributo: select
  Aplica las instrucciones de su interior para todos y cada uno de los nodos del conjunto de nodos dado por select
- Ejercicio: Al final del horario, sacar una lista de todas las tareas, indicando si la tarea en cuestión es por la mañana (acaba ANTES de las 12), por la tarde (empieza DESPUES de las 12), o al mediodía (toca a las 12 de cualquier manera)
  - horario8.xsl
  - NOTA: Usar // para recorrer las tareas
  - Ayuda:

```
<xsl:for-each select="¿TODAS LAS TAREAS?">
    <P>
      <xsl:value-of select="¿NOMBRE DE TAREA?"/>-
      <xsl:choose>
                <xsl:when test="hora-fin &lt; 12"> Por la mañana </xsl:when>
                <xsl:when test="hora-ini &gt; 12"> Por la tarde </xsl:when>
                <xsl:otherwise>Al mediodía</xsl:otherwise>
      </xsl:choose>
    </P>
</xsl:for-each>
```

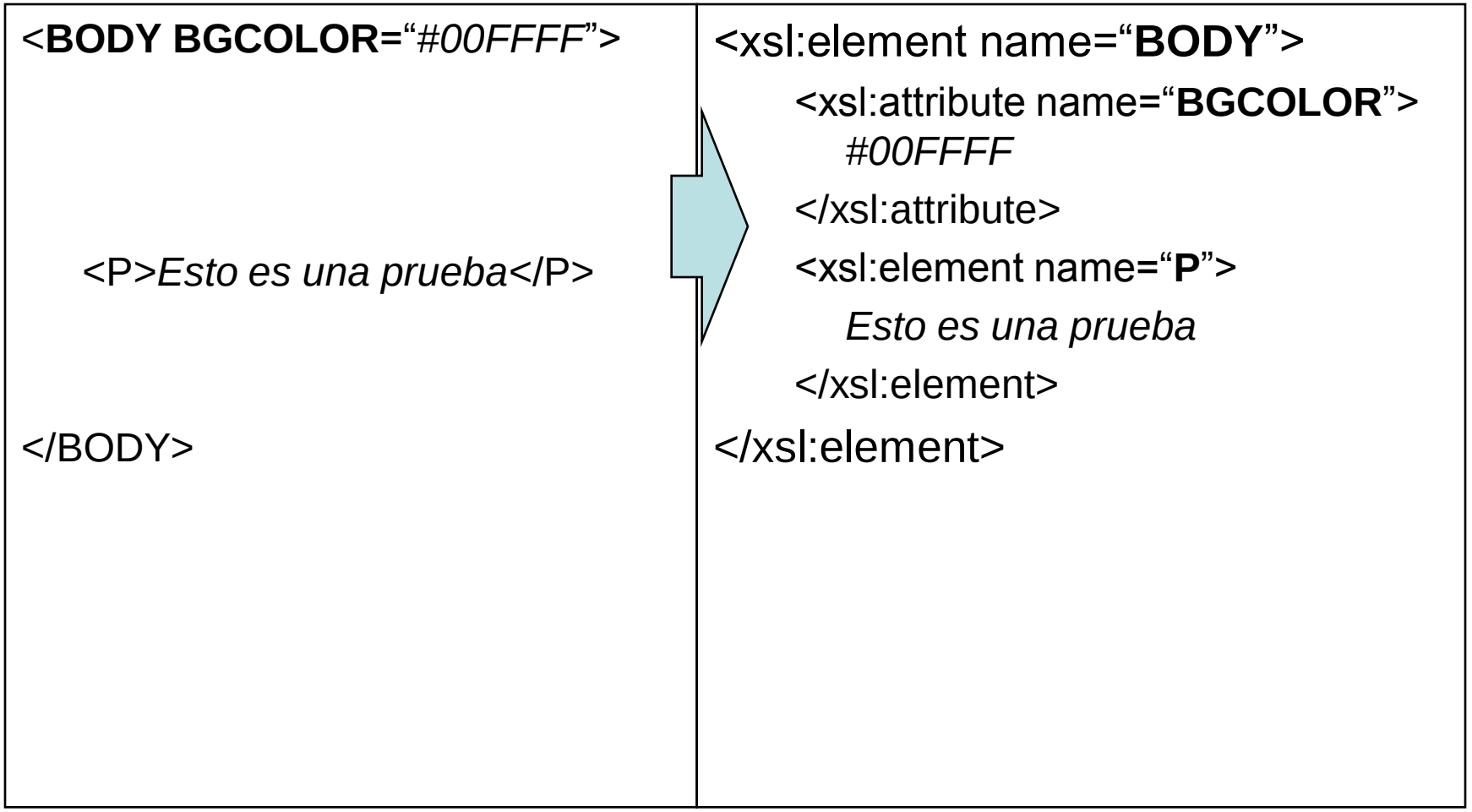
Instrucciones XSL

Construcción de elementos en el árbol resultado (I)

- ¿Cómo generar un elemento con cierta etiqueta y “construir” sus atributos?
- A veces la sintaxis no nos lo permite directamente. Posible ejemplo:
  - <BODY BGCOLOR=“<xsl:value-of select=“color-elegido”/>”>
- Se pueden utilizar los llamados AVT (Attribute Value Template): las expresiones entre llaves se evalúan como si hubiera un value-of
  - Para poner llaves "de verdad", poner cada una dos veces
- Se pueden necesitar instrucciones para “construir” dichos elementos
- xsl:element
  - Construcción de un elemento en el árbol resultado
  - Atributos: name
- xsl:attribute
  - Añadir un atributo al elemento en cuestión
  - Atributos: name
  - El valor está encerrado como texto libre dentro de xsl:attribute

Instrucciones XSL

Construcción de elementos en el árbol resultado (II)



Actividades finales

- **OBLIGATORIA**: Boletin de ejercicios "*EJERCICIOS XPATH.doc*" (incluye instrucciones precisas): Componer 27 expresiones xpath que respondan a cada una de las 27 cuestiones sobre datos de un XML dado (*tareas_finales_xpath.xml*)
- OPCIONALES:
  - Mejorar la hoja XSL para el horario: generar mejor HTML (título, por ejemplo)
  - Hacer que el horario salga en forma de tabla.
  - Hacer que salgan todos los días, aunque en el documento XML no estén.
  - Hacer que cada tarea salga en la casilla que ocupa en el horario.