Conversión y adaptación de documentos XML.

Caso práctico

La empresa de Félix y María continúa creciendo.

jonny goldstein - www.flickr.com (CC BY)

Dado que los documentos de los clientes de la empresa están en formato **XML**, Juan les plantea una nueva posibilidad para su página web. Dar la posibilidad a los clientes de consultar sus datos en la empresa a través de su página web.

Aunque a Félix y a María no les parece una idea muy seductora inicialmente, debido a que no consideran que el formato de los documentos **XML** sea el más atractivo para los clientes de la misma, deciden pedir consejo.

Una vez más, Juan les explica que existen unos lenguajes de marcas, en este caso **Xpath** y **XSL**, que les permitirán dar a la información de esos documentos el formato que quieran antes de publicarlo en una página segura de Internet.

Félix escucha atentamente las explicaciones de Juan sobre las posibilidades que da la transformación de documentos.

Gracias a ellas acaba de pensar que, además de permitir a los clientes consultar los datos a través de la web de la empresa, también puede servirles para generar informes en formato **PDF**, que pueden utilizar para informar a los clientes de sus negocios periódicamente, bien a través del correo ordinario o del electrónico.

María dice que empieza a creer que todo es posible mediante el uso de los lenguajes de marcas. Juan le dice que en lo que a intercambio de información se refiere si es así.Su pregunta ahora es, ¿de qué modo van a lograr separar la información de los documentos **XML** de las etiquetas de los mismos?Juan les explica que para eso existe un lenguaje llamado **Xpath**, cuya sintaxis es muy semejante a la que se usa para desplazarse a través de un árbol de directorios en modo comando.

1.- Introducción.

Caso práctico

Félix escucha atentamente las explicaciones de Juan sobre las posibilidades que da la transformación de documentos.

Gracias a ellas acaba de pensar que, además de permitir a los clientes consultar los datos a través de la web de la empresa, también puede servirles para generar informes en formato **PDF**, que pueden utilizar para informar a los clientes de sus negocios periódicamente, bien a través del correo ordinario o del electrónico.

Los documentos XML son documentos de texto con etiquetas, que contienen exclusivamente información, sin entrar en detalles de formato.

Por eso, si queremos usar directamente los datos (para leer, imprimir, etc.) es necesario transformar primero el documento **XML**.

Los navegadores, por ejemplo, interpretan las etiquetas del documento **XML**, les aplican un formato según lo especificado en las hojas **CSS** y lo muestran al usuario.

Es posible transformar un documento **XML** en otro tipo de documento. A esto se le denomina transformación de documentos. Algunas tecnologías que entran en juego en la transformación de documentos son:

- **XSLT**: permite definir el modo de transformar un documento **XML** en otro.
- **XSL-FO**: permite transformar un documetno **XML** en otro documente de un formato legible e imprimible por una persona, por ejemplo en un documento **PDF**.
- **XPath**: permite acceder a los diversos componentes de un documento **XML**.

Hoy en día se usa masivamente **XSLT,** que es ya un estándar aprobado por el **W3C.** Los documentos **XSLT** se denominan **hojas XSLT.**

XSLT es uno de los lenguajes derivados de **XML**, por tanto las hojas **XSLT** también son documentos **XML** (al igual que sucede con los canales **RSS**, **atom** o los documentos **XSD**).

¿Qué transformaciones podemos realizar sobre un documento **XML** usando **XSLT**? Podemos generar:

- Otro documento **XML**.
- Un documento **HTML**.
- Un documento de texto.

Autoevaluación

Los lenguajes de marcas que no permiten la transformación de documentos son:

- [ ] HTML.
- [ ] XHTML.
- [ ] XPath.
- [ ] XSL

Mostrar retroalimentación

Solución

1. Correcto
2. Correcto
3. Correcto
4. Incorrecto

2.- Estructura básica de una hoja XSLT.

¿Cuáles son los elementos contenidos en una hoja **XSLT**?

Básicamente hay tres tipos de elementos:

- **Elementos XSLT**, están precedidos del prefijo **`xsl:`**, pertenecen al espacio de nombres **`xsl`**, están definidos en el estándar del lenguaje y son interpretados por cualquier procesador **XSLT**.
- **Elementos LRE**, no pertenecen a **XSLT**, sino que se repiten en la salida sin más.
- **Elementos de extensión**, al igual que los anteriores, no pertenecen al espacio de nombres **`xsl`**. Son manejados por implementaciones concretas del procesador. Estos normalmente no son usados.

Y, ¿cómo indicamos que un documento **XML** está asociado a una hoja **XSLT**?

Hay que incluir en el documento **XML**, después del prólogo, una línea como la que sigue:

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

Autoevaluación

Cuáles de los siguientes tipos de documentos pueden obtenerse a partir de la transformación XSL de un documento XML:

- [ ] DOC.
- [ ] HTML.
- [ ] PDF.
- [ ] Texto.

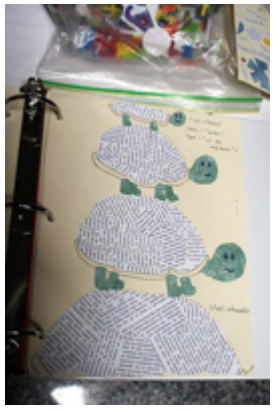
Mostrar retroalimentación

Solución

1. Incorrecto
2. Correcto
3. Incorrecto
4. Correcto

3.- Elementos XSLT.

El elemento raíz de una hoja **XSLT** es **`xsl:stylesheet`** o **`xls:transform`**, que son prácticamente equivalentes. Sus atributos principales son:



quinn.anya - www.flickr.com (CC BY-SA)

- **`version`**, cuyo valor puede ser 1.0 o 2.0.
- **`xmlns:xsl`**, se utiliza para declarar el espacio de nombres **`xsl`**. Para **XSLT** suele ser la dirección:

  http://www.w3.org/1999/XSL/Transform

A los elementos hijos de estos se les conoce como elementos de nivel superior, son estructuras contenedoras de instrucciones. Dado que si son hijos directos no pueden anidarse, excepto **`xsl:variable`** y **`xsl:param`**. Los más destacados son:

- **`xsl:attribute`**, añade un atributo a un elemento en el árbol de resultados.
- **`xsl:choose`**, permite decidir que parte de la hoja **XSL** se va a procesar en función de varios resultados.
- **`xsl:decimal-format`**, define un patrón que permite convertir en cadenas de texto números en coma flotante.
- **`xsl:for-each`**, aplican sentencias a cada uno de los nodos del árbol que recibe como argumento.
- **`xsl:if`**, permite decidir si se va a procesar o no una parte del documento **XSL** en función de una condición
- **`xsl:import`**, importa una hoja de estilos **XSLT** localizada en una **URI** dada.
- **`xsl:key`**, define una o varias claves que pueden ser referenciadas desde cualquier lugar del documento.
- **`xsl:output`**, define el tipo de salida que se generará como resultado.
- **`xsl:preserve-space`**, especifica cuales son los elementos del documento **XML** que no tienen espacios en blanco eliminados antes de la transformación.
- **`xsl:sort`**, permite aplicar un template a una serie de nodos ordenándolos alfabético numéricamente.
- **`xsl:strip-space`**, especifica cuáles son los elementos del documento **XML** que tienen espacios en blanco eliminados antes de la transformación.
- **`xsl:template`**, es el bloque fundamental de una hoja **XSLT**, por lo que veremos su descripción en el apartado siguiente.
- **`xsl:value-of`**, calcula el valor de una expresión **XPath** dada y lo inserta en el árbol de resultados del documento de salida.
- **`xsl:variable`**, asigna un valor a una etiqueta para usarlo cómodamente.

Autoevaluación

Selecciona los elementos que tienen esta funcionalidad. Debes elegir entre uno de estos:

1.- **preserve-space**
2.- **value-of**
3.- **strip-space**

4.- **decimal-format**

Debes introducir sólo el número de la opción seleccionada en cada caso. Ej: 1

Marca aquellos elementos que tienen espacios blancos eliminados antes de la transformación: ☐

Resuelve una expresión XPath dada: ☐

Marca los elementos que no tienen blancos eliminados: ☐

Convierte datos numéricos en cadenas de texto: ☐

Averiguar la puntuación | Mostrar retroalimentación | Mostrar/Eliminar las respuestas

Preserve-space marca los elementos que no tienen blancos eliminados, value-of resuelve una expresión XPath, strip-space marca los elementos que tienen algún espacio en blanco eliminado.

4.- XPath.

Caso práctico

María dice que empieza a creer que todo es posible mediante el uso de los lenguajes de marcas. Juan le dice que en lo que a intercambio de información se refiere si es así.Su pregunta ahora es, ¿de qué modo van a lograr separar la información de los documentos **XML** de las etiquetas de los mismos?Juan les explica que para eso existe un lenguaje llamado **Xpath**, cuya sintaxis es muy semejante a la que se usa para desplazarse a través de un árbol de directorios en modo comando.

XPath es un estándar (diferente de **XML**) aprobado por el **W3C**, que **permite navegar entre los elementos y atributos de un documento XML.**

Para hacerlo, se basa en las relaciones de parentesco entre los nodos del documento.

Inicialmente se creo para utilizarlo con **XLST**, pero en la actualidad se utiliza también con **`XML Schema`**, **`Xquery`**, **`Xlink`**, **`Xpointer`**, **`Xforms`**, etc.

Expresiones de camino

XPath se usa definiendo expresiones de camino, para seleccionar nodos o conjuntos de nodos en un documento **XML**.

Esas expresiones se parecen mucho a las expresiones de camino (**path**) que se suelen usar en los sistemas de ficheros.

Estas expresiones se aplican a un documento **XML**, asumiendo que su estructura interna es la de un árbol. Al aplicar una expresión, se obtiene un conjunto de nodos (que puede ser vacío).

En el siguiente código **XML** de ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<matriculas>
  <alumno>
    <nombre>Pedro</nombre>
    <apellido1>López</apellido1>
    <apellido2>Ortega</apellido2>
    <DNI pais="es">López</DNI>

```

```
9     </alumno>
  </matriculas>
```

Al evaluar la expresión **"/matriculas/alumno/nombre"** podríamos obtener los nombres de los alumnos matriculados.

Debes conocer

En el siguiente enlace puedes encontrar el estándar de Xpath que aprobó el W3C el 16 de Noviembre de 1999 y que ha evolucionado hasta el 2017.

Recomendación del W3C sobre XPath.

Autoevaluación

XPath es un lenguaje XML que permite:

- ○ Transformar el formato de los datos de un fichero XML.
- ○ Definir un vocabulario que ha de cumplir un documento XML.
- ○ Obtener los datos del fichero XML de una base de datos.
- ○ Acceder a los datos de un fichero XML.

No es correcta porque eso lo hace XSL.

Incorrecta, porque de ello se encarga XML Schema.

No es la respuesta correcta porque ningún lenguaje de marcas visto hasta ahora lo permite.

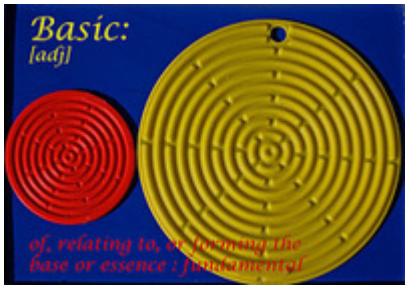
Muy bien. Has captado la idea.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

4.1.- Términos básicos.

Nodos



Un documento **XML** es tratado en **XPath** como un árbol de nodos. Hay 7 tipos de nodos: **elemento, atributo, texto, espacio de nombres, instrucción de proceso, comentario** y **documento**. El elemento más alto del árbol es el **nodo raíz**. Explicamos más sobre algunos de ellos:

- **Nodo raíz**, es el nodo que contiene al ejemplar del fichero **XML**.
  - ¡Cuidado! No lo confundamos con el elemento raíz del documento, ya que éste último está por debajo de él.
- **Nodos elemento**, son cada uno de los elementos del documento **XML**.
  - Tienen un elemento padre.
  - El padre del elemento raíz, es decir del ejemplar, es el nodo raíz del documento.
  - Pueden tener identificadores únicos, lo que permite referenciarlos de forma mucho más directa. Para ello es necesario que el atributo esté definido en un **DTD** o un fichero **XSD** asociado.
- **Nodos texto**, son aquellos caracteres del documento que no están marcados con ninguna etiqueta.
  - No tienen hijos.
- **Nodos atributo**, son los atributos de un elemento.
  - Se consideran etiquetas añadidas al nodo elemento.
  - No se consideran hijos de ese elemento.
  - Aquellos atributos que tengan un valor asignado en el esquema asociado, se tratarán como si ese valor se le hubiese dado al escribir el documento **XML**.
  - Para las definiciones de los espacios de nombre y para aquellos atributos que se han definido con la propiedad **`#IMPLIED`** en su **DTD** no se crean nodos.

En el ejemplo de antes:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<matriculas>
  <alumno>
    <nombre>Pedro</nombre>
    <apellido1>López</apellido1>
    <apellido2>Ortega</apellido2>
    <DNI pais="es">López</DNI>
  </alumno>
</matriculas>
```

- **`<matrículas>`** es el nodo elemento raíz
- **`<nombre>Pedro</nombre>`** es un nodo elemento
- **`pais="es"`** es un nodo atributo

Ítems

Los **ítems** pueden ser nodos o valores atómicos.
En el ejemplo, valores atómicos serían "Pedro", o "es"

Relaciones entre nodos

Según como si sitúen los nodos dentro del árbol, se habla de las relaciones entre ellos. Así se dice que hay relaciones padre-hijo, hermano, ascendentes y descendentes.
En ejemplo anterior, algunas de estas relaciones son:

- **<nombre>** es hijo de **<alumno>**.
- **<nombre>** y **<DNI>** son hermanos.
- **<matriculas>** es un ascendiente de **<nombre>**
- **<apellido1>** es descendiente de **<matriculas>**

Autoevaluación

El nodo raíz de un documento XML coincide con el ejemplar del mismo:

- ○ Sí.
- ○ No.

No es correcta porque el nodo raíz contiene el ejemplar del documento.

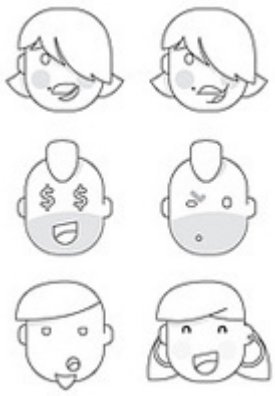
Efectivamente es correcto, es importante que no confundas el ejemplar con el nodo raíz de un documento.

Solución

1. Incorrecto
2. Opción correcta

4.2.- Expresiones.

Como hemos dicho, **XPath usa expresiones, que serán evaluadas.**



Florin Hatmanu - www.flickr.com
(CC BY-NC-ND)

¿Y cuáles son los resultados que da la evaluación de una expresión XPath?

Pues podemos obtener cuatro tipos de resultados diferentes:

- Un conjunto de nodos (node-set)
  - No está ordenado.
  - Se considera que todos los elementos de un conjunto de nodos son hermanos, independientemente de lo que fuesen originalmente.
  - Aunque los hijos de los nodos que forman un conjunto de nodos son accesibles, los subárboles de un nodo no se consideran elementos del conjunto.
- Un valor booleano.
- Un número.
- Una cadena.

¿Qué elementos podemos utilizar en una expresión XPath?

Podemos utilizar :

- Agrupaciones: “()”, “{}”, “[ ]”.
- Elemento actual, elemento padre.
- Atributos: “@”.
- Elementos, “*”.
- Separadores, “::”.
- Comas, “,”.
- El nombre de un elemento.
- Tipo de nodo, que puede ser:
  - **`comment`**.
  - **`text`**.
  - **`procesing instruction`**.
  - **`node`**.
- Operadores: **`and, or, mod, div, *, /, //, |, +, -, =, !=, <, >, <=, >=.`**
- Nombres de función.
- Denominación de ejes: **`ancestor, ancestor-or-self-atribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self.`**
- Literales, se ponen entre comillas dobles o simples. Pueden anidarse alternando el tipo de comillas.
- Números.
- Referencias a variables, para lo que se utiliza la sintaxis: **`$nombreVariable.`**

Para ver cómo se emplean cada uno de ellos, puedes encontrar muy buenos ejemplos aquí.

Autoevaluación

Indica cuáles de los siguientes elementos de un documento XML pueden ser nodos del mismo:

- [ ] Atributos.
- [ ] Comentarios.
- [ ] Etiquetas.
- [ ] Texto.

Mostrar retroalimentación

Solución

1. Correcto
2. Correcto
3. Incorrecto
4. Correcto

4.3.- Procesando expresiones.

Cuando estamos procesando un documento **XML** de entrada, podemos hablar de los siguientes conceptos:

- **Nodo actual**, es aquél en el que se encuentra el procesador.
- **Nodo contexto**, cada expresión está formada por subexpresiones que se van evaluando antes de resolver la siguiente. El conjunto de nodos obtenido tras evaluar una expresión y que se utiliza para evaluar la siguiente es el nuevo contexto.
- **Tamaño del contexto**, es el número de nodos que se están evaluando en un momento dado en una expresión XPath.

4.4.- Ruta de Localización.

La **ruta de localización de un nodo** es la ruta que hay que seguir en un árbol de datos para localizar ese nodo. Las rutas de localización se evalúan y esa evaluación siempre **devuelve un conjunto de nodos**, aunque puede estar vacío.

¿Cómo podemos crear una ruta de localización? Mediante la unión de varios pasos de localización.

Maxim BA - www.flickr.com (CC BY-NC-ND)

Las rutas de localización básicas son:

- **La ruta de localización del nodo raíz del documento**. Es la barra diagonal "/". Se trata de una ruta absoluta, porque siempre significa lo mismo, independientemente de la posición del procesador en el documento de entrada al aplicar la regla.
- **Localización de un elemento**, selecciona todos los hijos del nodo de contexto con el nombre especificado. Los elementos a los que se refiera dependerán de la localización del nodo de contexto, por lo que es una ruta relativa. En el caso de que el nodo contexto no tenga ningún nodo hijo con esa denominación, el valor dela ruta de localización será un elemento vacío.
- **Localización de atributos**, para referirnos a ellos en **XPath**, se utiliza el símbolo "@" seguido del nombre del atributo deseado.
- **Localización de espacios de nombres**, no se tratan explícitamente.
- **Localización de comentarios**.
- **Localización de nodos de texto**.
- **Localización de instrucciones de procesamiento**.

¿Cómo podemos comparar distintos elementos y tipos de nodo simultáneamente? Utilizando alguno de los tres comodines mostrados a continuación:

- **Asterisco** "*": compara cualquier nodo de elemento, independientemente de su nombre. No compara ni atributos ni comentarios, nodos de texto o instrucciones de procesamiento.
- **`Node()`**: compara, además de los tipos de elementos, el nodo raíz, los nodos de texto, los de instrucción de procesamiento, los nodos de espacio de nombre, los de atributos y los de comentarios.
- **`"@*"`**: compara los nodos de atributo.

Autoevaluación

Las rutas de localización:

- ○ Devuelven todos los nodos de un documento XML que verifican una condición dada.
- ○ Devuelven todos los nodos de un árbol XML que verifican una condición dada.

- ○ El primer nodo que se encuentra y cumple una condición dada.
- ○ No pueden devolver valores de atributos.

No es correcta porque el resultado dependerá del nodo que usemos como origen de la ruta.

Muy bien. Esto lo has entendido, pasa al siguiente apartado.

No es la respuesta correcta, porque devuelve TODOS los nodos de la rama que evalúa que se verifica la condición.

No es correcta, ya que los atributos son un nodo más y por tanto son tratados como tales.

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

4.5.- Funciones de XPath.

XPath también nos proporciona funciones.

Podemos emplearlas, unidas a las rutas de localización, para hacer cosas sobre los conjuntos de elementos que devuelven los predicados (que veremos enseguida).

Entre las funciones más importantes que podemos utilizar en **XPath** destacan:

vestman - www.flickr.com (CC BY)

- **`boolean()`**, al aplicarla sobre un conjunto de nodos devuelve true si no es vacío.
- **`not()`**, al aplicarla sobre un predicado devuelve true si el predicado es falso , y falso si el predicado es true.
- **`true()`**, devuelve el valor true.
- **`false()`**, devuelve el valor false.
- **`count()`**, devuelve el número de nodos que forman un conjunto de nodos.
- **`name()`**, devuelve un nombre de un nodo.
- **`local-name()`**, devuelve el nombre del nodo actual o del primer nodo de un conjunto de nodos.
- **`namespace-uri()`**, devuelve el **URI** del nodo actual o del primer nodo de un conjunto dado.
- **`position()`**, devuelve la posición de un nodo en su contexto comenzando en 1. Por ejemplo, para seleccionar los dos primeros elementos de tipo elemento de un fichero **XML** pondremos: **`//elemento[position()<=2]`**
- **`last()`**, Devuelve el último elemento del conjunto dado.
- **`normalize-space()`**, permite normalizar los espacios de una cadena de texto, es decir, si se introduce una cadena donde hay varios espacios consecutivos, esta función lo sustituye por uno solo.
- **`string()`**, es una función que convierte un objeto en una cadena. Los valores numéricos se convierten en la cadena que los representa teniendo en cuenta que los positivos pierden el signo. Los valores *booleanos* se convierten en la cadena que representa su valor, esto es “true” o “false”.
- **`concat()`**, devuelve dos cadenas de texto concatenadas. El ejemplo siguiente devuelve “**XPath** permite obtener datos de un fichero **XML**”.

```
concat('XPath', 'permite obtener datos de un fichero XML')
```

- **`string-length()`**, devuelve la cantidad de caracteres que forman una cadena de caracteres.
- **`sum()`**, devuelve la suma de los valores numéricos de cada nodo en un conjunto de nodos determinado.

Autoevaluación

Para lograr ir directamente al último de los elementos hijos de un elemento en el que nos encontramos habrá que usar la función:

- ○ position(end)
- ○ last()
- ○ first()
- ○ end()

No es correcta porque la expresión dada no existe.

¡Genial! Continúa así.

No es la respuesta correcta, porque queremos el último elemento y este nos daría el primero de la rama en la que estemos situados.

Incorrecta end no es una función en Xpath.

Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

4.6.- Predicados.

Un predicado es una expresión booleana que añade un nivel de verificación al paso de localización.

En estas expresiones podemos incorporar funciones **XPath**.

¿Qué ventajas nos da el uso de predicados?

Mediante las rutas de localización se pueden seleccionar varios nodos a la vez, pero el uso de predicados permite seleccionar un nodo que cumple ciertas características.

Los predicados se incluyen dentro de una ruta de localización utilizando los corchetes, por ejemplo:

```
/receta/ingredientes/ingrediente[@codigo="1"]/nombre
```

En este caso se está indicando al intérprete que escoja, dentro de un fichero **XML** de recetas, el nombre del ingrediente cuyo código tiene el valor "1".

Veamos que es lo que podemos incluir en un predicado.

- **Ejes**, permiten seleccionar el subárbol dentro del nodo contexto que cumple un patrón. Pueden ser o no de contenido.
  - **`child`**, es el eje utilizado por defecto. Su forma habitual es la barra, **"/"**, aunque también puede ponerse: **`/child::`**
  - **`attribute`**, permite seleccionar los atributos que deseemos. Es el único eje que veremos que no es de contenido.
  - **`descendant`**, permite seleccionar todos los nodos que descienden del conjunto de nodos contextos. Se corresponde con la doble barra, **"//"**, aunque se puede usar: **`descendant::`**
  - **`self`**, se refiere al nodo contexto y se corresponde con el punto **"."**.
  - **`parent`**, selecciona los nodos padre, para referirnos a él usamos los dos puntos, **".."**.
  - **`ancestor`**, devuelve todos los nodos de los que el nodo contexto es descendiente.
- **Nodos test**, permiten restringir lo que devuelve una expresión **XPath**. Podemos agruparlos en función de los ejes a los que se puede aplicar.
  - Aplicable a cualquier eje:
    - **"*"**, solo devuelve elementos, atributos o espacios de nombres pero no permite obtener nodos de texto, o comentarios de cualquier tipo.
    - **`nod()`**, devuelve los nodos de todos los tipos.
  - Solo aplicables a ejes de contenido:
    - **`text()`**, devuelve cualquier nodo de tipo texto.
    - **`comment()`**, devuelve cualquier nodo de tipo comentario.
    - **`processing-instruction()`**, devuelven cualquier tipo de instrucción de proceso.

Varios predicados pueden unirse mediante los operadores lógicos **`and`**, **`or`** o **`not`**.

Autoevaluación

Relaciona los ejes siguientes con sus equivalentes.Debes elegir entre las siguientes representaciones y escribirlas en el hueco correspondiente: **@** , **/** , **//** , **..**

El eje child:: se representa por ☐

El eje descendant:: se representa por ☐

El eje atribute:: se representa por ☐

El eje parent:: se representa por ☐

Enviar

Child:: se representa por “/”, descendant:: por “//”, attribute por “@”, parent:: por “..”.

4.7.- Ejemplo - Ejercicio.

Ejercicio Resuelto

Dado el siguiente fichero XML

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!DOCTYPE agenda>
<agenda>
  <propietario>
    <identificadores>
      <nombre>Alma</nombre>
      <apellidos>López Terán</apellidos>
    </identificadores>
    <direccion>
      <calle>El Percebe 13, 6F</calle>
      <localidad>Torrelavega</localidad>
      <cp>39300</cp>
    </direccion>
    <telefonos>
      <movil>970898765</movil>
      <casa>942124567</casa>
      <trabajo>628983456</trabajo>
    </telefonos>
  </propietario>
  <contactos>
    <persona id="p01">
      <identificadores>
        <nombre>Inés</nombre>
        <apellidos>López Pérez</apellidos>
      </identificadores>
      <direccion>
        <calle>El Ranchito 24, 6B</calle>
        <localidad>Santander</localidad>
        <cp>39006</cp>
      </direccion>
      <telefonos>
        <movil>970123123</movil>
      </telefonos>
    </persona>
    <persona id="p02">
      <identificadores>
        <nombre>Roberto</nombre>
        <apellidos>Gutiérrez Gómez</apellidos>
      </identificadores>
      <direccion>
        <calle>El Marranito 4, 2F</calle>
        <localidad>Santander</localidad>
        <cp>39004</cp>
      </direccion>
```

```
      <telefonos>
        <movil>970987456</movil>
        <casa>942333323</casa>
      </telefonos>
    </persona>
    <persona id="p03">
      <identificadores>
        <nombre>Juan</nombre>
        <apellidos>Sánchez Martínez</apellidos>
      </identificadores>
      <direccion>
        <calle>El Cangrejo 10, sn</calle>
        <localidad>Torrelavega</localidad>
        <cp>39300</cp>
      </direccion>
      <telefonos>
        <movil>997564343</movil>
        <casa>942987974</casa>
        <trabajo>677899234</trabajo>
      </telefonos>
    </persona>
  </contactos>
</agenda>
```

Construir las sentencias **XPath** que permitan obtener los siguientes datos:

1.- Nombre del propietario de la agenda.
2.- Teléfono de casa del propietario.
3.- Nombres y apellidos de los contactos de la agenda.
4.- Nombre e identificador de cada contacto.
5.- Datos del contacto con identificador "p02".
6.- Identificadores de los contactos que tienen móvil.

Te facilitamos el fichero XML en el siguiente enlace para tu comodidad en la resolución del ejercicio:

Mostrar retroalimentación

- Nombre del propietario de la agenda.

  /agenda/propietario/identificadores/nombre

- Teléfono de casa del propietario.

  /agenda/propietario/telefonos/casa

- Nombres y apellidos de los contactos de la agenda.

  //contactos/persona/identificadores/nombre | //contactos/persona/identificadores/apellidos

- Nombre e identificador de cada contacto.

```
//contactos/persona/identificadores/nombre      |
//contactos/persona/@id
```

- Datos del contacto con identificador "p02".

```
//contactos/persona[@id="p02"]/*/*
```

- Identificadores de los contactos que tienen teléfono en casa.

```
//contactos/persona/telefonos/casa/../../@id
```

4.9.- Acceso a datos de otro documento XML.

Además de acceder a los datos del fichero **XML** con el que se está trabajando directamente, es útil poder acceder a los datos de otros ficheros.

Para ello utilizaremos la función **`document()`**, pero dicha función NO es del lenguaje **XPath**, sino que pertenece a **XSLT**.

Esta función puede admitir dos argumentos diferentes:

`document(URI)`

En este caso la función devuelve el elemento raíz del documento **XML** que se localiza en el **URI** especificado.

`document(nodo)`

En esta ocasión lo que devuelve la función, es el conjunto de nodos cuya raíz es el nodo dado.

Autoevaluación

Indica cuál de estas afirmaciones es correcta:

- ○ La función document() devuelve únicamente una rama de un nodo dado.
- ○ La función document() devuelve únicamente una rama de un nodo dado.
- ○ No existe ningún modo de acceder a los elementos de varios documentos.
- ○ XPath no proporciona un modo de acceder a datos de varios documentos simultáneamente.

No es correcta porque document no es una función de Xpath.

Incorrecta porque esta función también devuelve el árbol de nodos de un documento dada su localización.

No es la respuesta correcta porque XSL si lo permite.

Efectivamente es correcto, document() no es una función de Xpath.

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

5.- Utilización de plantillas.

El elemento `xml:template`

El elemento **`xsl:template`** permite controlar el formato de salida que se aplica a ciertos datos de entrada.

Tiene un atributo denominado **`match`**, que se utiliza para seleccionar los nodos del árbol de entrada (conforme a **XPath**) a los que se va a aplicar la plantilla.

Para especificar el formato de salida, se emplean sentencias **XHTML** (el contenido del elemento **`template`** ha de ser **XHTML** bien estructurado).

Para especificar el punto de la salida donde queremos que se apliquen las plantillas, se emplea el elemento **`xsl:apply-templates`**.

Tiene un atributo **`select`**, que se utiliza para seleccionar los hijos del nodo de entrada (conforme a **XPath**) a los que se va a aplicar la plantilla.

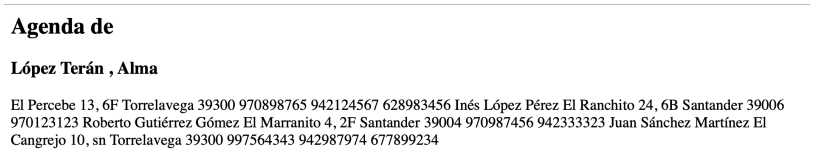
Podemos usar ese atributo para especificar el orden en que van a ser procesados los nodos hijo. Si no lo hacemos de este modo, se aplicarán en el orden es el que el interprete utiliza al leer el documento **XML**, es decir, de arriba abajo.

Ejemplo de uso

Usando el código **XML** visto en el *apartado 4.7*, podríamos usar el siguiente código:

```
<xsl:template match="/">
<html>
  <body>
    <h2>Agenda de </h2>
    <xsl:apply-templates/>
  </body>
</html>
</xsl:template>
<xsl:template match="propietario/identificadores">
<h3>
<xsl:apply-templates select="apellidos"/>
,
<xsl:apply-templates select="nombre"/>
</h3>
</xsl:template>
```

Generaríamos una salida formateada como la siguiente:



Con la primera plantilla creamos el documento **`html`** y el **`body`** y ponemos el título principal.

Con la segunda, seleccionamos al propietario de la agenda y lo formateamos como encabezado 3.

5.1.- Ejercicio resuelto de Plantillas.

Ejercicio Resuelto

Dada la siguiente plantilla **XSLT** (disponible en formato fichero en este enlace):

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version=

<xsl:template match="identificadores">
<xsl:value-of select="nombre"/>,
<xsl:value-of select="apellidos"/>
</xsl:template>

<xsl:template match="persona">
<xsl:apply-templates select="identificadores"></xsl:apply-templates>
</xsl:template>
</xsl:stylesheet>
```

Que se desea aplicar sobre el siguiente fichero (puedes descargarlo haciendo clic aquí):

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!DOCTYPE agenda>
<?xml-stylesheet type="text/xsl" href="./LMSGI_CONT_Ejemplo02.xsl"?>
<agenda>
<persona id="p01">
<identificadores>
<nombre>Inés</nombre>
<apellidos>López Pérez</apellidos>
</identificadores>
<direccion>
<calle>El Ranchito 24, 6B</calle>
<localidad>Santander</localidad>
<cp>39006</cp>
</direccion>

<telefonos>
<movil>970123123</movil>
</telefonos>
</persona>

<persona id="p02">
<identificadores>
```

```
<nombre>Roberto</nombre>
<apellidos>Gutiérrez Gómez</apellidos>
</identificadores>

<direccion>
<calle>El Marranito 4, 2F</calle>
<localidad>Santander</localidad>
<cp>39004</cp>
</direccion>

<telefonos>
<movil>970987456</movil>
<casa>942333323</casa>
</telefonos>
</persona>
</agenda>XML
```

Indicar cuál sería el **resultado de aplicar la transformación XSLT sobre el fichero XML**.

Mostrar retroalimentación

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

Inés,
López Pérez

Roberto,
Gutiérrez Gómez

Juan,
Sánchez Martínez
```

6.- Procesadores XSLT.

Un procesador **XSLT** es un software que lee un documento **XSLT** y otro **XML**, y crea un documento de salida aplicando las instrucciones de la hoja de estilos **XSLT** a la información del documento **XML**.

Rubén Marcos - www.flickr.com (CC BY-NC-ND)

Pueden estar integrados dentro de un explorador Web, en un servidor web, o puede ser un programa que se ejecuta desde la línea de comandos.

- Existen diferentes modos de realizar la transformación **XSLT**
- Mediante el procesador *MSXML*, (servicios principales de **Microsoft XML**).
- Usando un procesador *XSLTPROC*, por ejemplo **`xsltproc`** desde línea de comandos.
- Invocando a la biblioteca de transformación desde un programa.
- Realizando un enlace entre la hoja **XSLT** y el documento **XML**, en este caso hay que añadir, en el fichero **XML** entre el la definición de la versión **XML** y la definición del tipo de documento, la línea:

```
<?xml-stylesheet type="text/xsl" href="path_hoja_xsl"?>
```

El fichero puede verse directamente desde cualquier navegador que soporte **XSLT**, aunque tiene la desventaja de que queda ligado a esa vista.
La mayoría de editores **XML** permiten escoger el interprete que debe encargarse de procesar un documento **XSLT**.

Autoevaluación

El único modo de generar un documento a partir de una transformación XSLT de otro es utilizando un editor XML que tenga incorporado un procesador XSLT:

- ○ No.
- ○ Sí.

Muy bien, has captado la idea.

Incorrecta, podemos utilizar un procesador en modo comando, sin necesidad de editor, o un procesador integrado en un navegador.

Solución

1. Opción correcta
2. Incorrecto

Para saber más

En los siguientes enlaces puedes encontrar algunos de los procesadores de XSLT:

Xalan

SAXON

7.- Depuradores XSLT.

Los depuradores son elementos de software que permiten seguir la generación de un documento, a partir de los datos de un documento **XML** aplicándoles una hoja de estilo **XSLT**.

Franz Patzig - www.flickr.com (CC BY-NC-SA)

Los depuradores **XSLT** nos facilitan la localización y corrección de errores dejándonos ejecutar el código paso a paso, salir, ejecutar el código hasta el cursor, ejecutar hasta el final, pausar y detener.

Algunos editores de **XML** (sobretodo los comerciales), incluyen un depurador que permite visualizar simultáneamente la plantilla que se está ejecutando y sobre qué datos del documento **XML** actúan y cuál es la salida que genera dicha orden.

De este modo es más fácil averiguar cuáles son las plantillas que dan lugar al formato que deseamos en la salida del documento.

Para saber más

En el siguiente vídeo se muestra cómo ejecutar y depurar una transformación XSL utilizando el editor Oxygen XML Editor:

Ejecución y depuración de ficheros XSL.

https://www.youtube.com/embed/gwYGEY80vi4

Resumen textual alternativo

8.- Para saber más.

Para saber más

Información adicional:

- Recomendación del W3C sobre XSLT version 1.0.
- Recomendación del W3C sobre XSLT version 1.1.
- Recomendación del W3C sobre XSLT version 2.0.
- Recomendación del W3C sobre XSLT versión 3.0.
- MCLibre - teoría y ejemplos XPath