LMSGI _05

Conversión y adaptación de documentos XML

Conversión y adaptación de documentos XML.

1.- Técnicas de transformación de documentos XML.

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. Esto implica la necesidad de algún medio para expresar la transformación de un documento XML, para que una persona pueda utilizar directamente los datos para leer, imprimir, etc.

Las 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**: Se utiliza para transformar XML en un formato legible e imprimible por una persona, por ejemplo en un documento PDF.
- **Xpath**: permite el acceso a los diversos componentes de un documento XML

La parte de transformaciones ganó en importancia, y se llega a la terminología actual que es XSL y comprende las anteriores.

Autoevaluación

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

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

2.- 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.

Una expresión XPath es un predicado que se aplica sobre una estructura de árbol correspondiente a la jerarquía de los datos de un documento XML, y devuelve todo lo que encaja con ese predicado.

Es un estándar diferente de XML, aprobado por el W3C, que nos permite acceder a partes de un documento XML basándonos en las relaciones de parentesco entre los nodos del documento.

Su notación es similar a las de las rutas de los ficheros.

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

Autoevaluación

XPath es un lenguaje XML que permite:

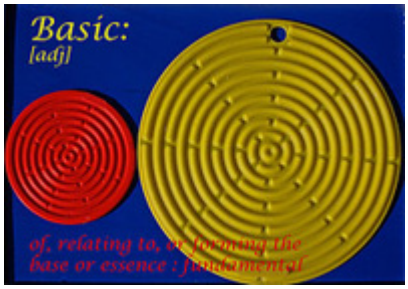
- Transformar el formato de los datos de un fichero XML.
- Definir un vocabulario que ha de cumplir un documento XML.
- Obtener los datos del fichero XML de una base de datos.
- Acceder a los datos de un fichero XML.

Debes conocer

En el siguiente enlace puedes encontrar el estándar de Xpath que aprobó el W3C el 19 de Noviembre de 1999:

Recomendación del W3C sobre XPath.

2.1.- Términos básicos.



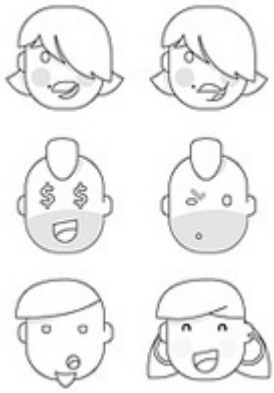
- **Nodo raíz**, es el nodo que contiene al ejemplar del fichero XML. No debe confundirse con el elemento raíz del documento, ya que éste último está por debajo de él.
- Se identifica por "/".
- **Nodos elemento**, son cada uno de los elementos del documento XML. Todos ellos tienen un elemento padre, el padre del elemento raíz, es decir del ejemplar, es el nodo raíz del documento.
- Pueden tener identificadores únicos, para ello es necesario que un atributo esté definido de ese modo en un DTD o un fichero XSD asociado, esto nos permite referenciar dicho elemento de forma mucho más directa.
- **Nodos texto**, son aquellos caracteres del documento que no están marcados con ninguna etiqueta. Este tipo de nodos no tienen hijos.
- **Nodos atributo**, no se consideran hijos del elemento al que están asociados sino etiquetas añadidas al nodo 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.
- **Nodos de comentario y de instrucciones de proceso**, son los nodos que se generan para los elementos con comentarios e instrucciones de proceso.
- **Nodo actual**, es aquél al que nos referimos cuando se evalúa una expresión en Xpath.
- **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.

Autoevaluación

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

- Sí.
- No.

2.2.- Expresiones.



¿Cuáles son los resultados que da la evaluación de una expresión Xpath?

Podemos obtener cuatro tipos de resultados diferentes:

- Un conjunto de nodos que no está ordenado, **node-set.**. 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.
  Los nodos pueden ser de 7 tipos diferentes:
  - Elemento.
  - Atributo.
  - Texto.
  - Espacio de nombres.
  - Instrucción de procesamiento.
  - Comentario.
  - Raíz.
- **Booleano.**
- **Número.**
- **Cadena.**

¿Cuáles son los tokens que podemos utilizar en una expresión XPath?

- Paréntesis, “()”; llaves , “{}” y corchetes, “[ ]”.
- Elemento actual, elemento padre.
- Atributo, “@”.
- Elemento, “*”.
- Separador, “::”.
- Coma, “,”.
- 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**

Autoevaluación

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

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

2.3.- Ruta de localización.

Se corresponde con la ruta que hay que seguir en un árbol de datos para localizar un nodo.
La evaluación de una ruta de localización 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.

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.

2.4.- 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.

Ejercicio de relacionar.

Eje.	Relación.	Representación.
child::	☐	1. @
descendant::	☐	2. /
attribute::	☐	3. //
parent::	☐	4. ..

2.5.- Funciones de Xpath.

Entre las funciones más importantes que podemos utilizar en Xpath destacan:

- **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()

2.6.- Acceso a atributos.

¿Cómo podemos acceder a los atributos?

Utilizando el eje **attribute**:, que puede sustituirse por la arroba, "**@**".

Ejercicio resuelto

Dado el siguiente fichero 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:

2.7.- Acceso a elementos 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:

- Para aplicar una ruta de XPath a un documento diferente de aquel con el que se trabaja en un momento dado el lenguaje nos proporciona la función document().
- 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.

3.- XSLT.

XSLT es un estándar aprobado por el W3C, ¿pero una hoja XSLT es también un documento XML?

Sí, 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?

- A otro documento XML.
- A un documento HTML.
- A un documento de texto.

Debes conocer

En los siguientes enlaces puedes encontrar el estándar de las diferentes versiones que existen de XSLT aprobadas por el W3C

Recomendación del W3C sobre XSLT version 1.0.

Recomendación del W3C sobre XSLT version 1.1.

Recomendación del W3C sobre XSLT version 2.0.

3.1.- 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.

Ahora que sabemos como es la estructura de un fichero XSLT, nos preguntamos si para asociar un fichero XML a un XSLT hay que incluir en ellos algún elemento, como se hace con los vocabularios.

La respuesta es afirmativa. Para indicar que un fichero XML está asociado a un XSLT hay que añadir, después del prólogo, una línea como la que sigue:

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

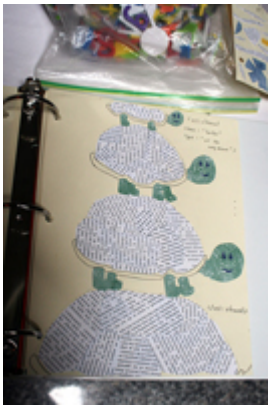
Con ella se indica al procesador el lugar donde se encuentra la hoja de estilos con la que dar formato a los datos del documento.

Autoevaluación

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

- [ ] doc.
- [ ] HTML.
- [ ] PDF.
- [ ] Texto.

3.2.- Elementos XSLT.



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

- **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 cuales 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

Relaciona los siguientes elementos con sus funcionalidades.

Ejercicio de relacionar.

Elemento	Relación	Funcionalidad
preserve-space.	☐	1. Convierte datos numéricos en cadenas de texto.
value-of	☐	2. Marca aquellos elementos que tienen espacios blancos eliminados antes de la transformación.
strip-space	☐	3. Resuelve una expresión XPath dada.
decimal-format	☐	4. Marca aquellos elementos que no tienen espacios blancos eliminados antes de la transformación.

3.3.- Utilización de plantillas.

El elemento **xsl:template** permite controlar el formato de salida que se aplica a ciertos datos de entrada. Dicho formato se especifica utilizando sentencias XHTML

Tiene un atributo denominado **match**, que se utiliza para seleccionar los nodos del árbol de entrada.

Un ejemplo simple del uso de este elemento es, siempre utilizando como fuente de datos el fichero XML que se ha descrito en el apartado 2.6 de este tema:

```
<xsl:template match=”propietario”>
<p>Agenda de </p>
</xsl:template>
```

Donde el elemento < p> es el elemento párrafo de XHTML.
Hay que tener en cuenta que el contenido del elemento template ha de ser XHTML bien estructurado.

En una plantilla podemos tener más de una regla que afecte a distintos elementos. ¿Cuál es el orden de aplicación de estas reglas de formato? Por defecto ese orden es el que el interprete utiliza al leer el documento XML, es decir, de arriba abajo, aunque dicho orden puede ser modificado en la plantilla, para ello utilizaremos el elemento **xsl:apply-templates**. Un ejemplo del uso es:

```
<xsl:template match=”contactos”>
<xsl:apply-templates select=”identificadores”/>
</xsl:template>
```

Con esto la plantilla solo se aplicará sobre los datos incluidos dentro del elemento identificadores y no sobre los elementos *direccion* o *telefonos*.

Ejercicio resuelto

Veamos un ejemplo de una hoja de estilos completa.

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">

<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>
```

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

Ejemplo02.xsl

Cuando un procesador XSLT aplica esta hoja de estilos al fichero XML siguiente:

```
<?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>
```

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

Ejemplo02.xml

3.4.- 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.

Pueden estar integrados dentro de un explorador Web, como MSXML en IE 6, en un servidor web, como el Cocoon de Apache XML o puede ser un programa que se ejecuta desde la línea de comandos, como Xalan de Apache o SAXON.

Existen diferentes modos de realizar la transformación XSLT

- Mediante el procesador MSXML, que es un ejecutable que se limita a llamar a la biblioteca de transformación de Internet Explorer.
- Usando un procesador XSLTPROC..
- 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, como Oxygen, Editix o Estudio XML Líquido 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.

3.5.- Depuración.

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.

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.

Los editores de XML, como Oxygen XML o Editix, 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 se facilita averiguar cuales son las plantillas que no dan lugar al formato que deseamos en la salida del documento.