3 Lenguajes para el almacenamiento y transmisión de información

OBJETIVOS DEL CAPÍTULO

- ✓ Conocer los tipos de lenguajes para el almacenamiento y transmisión de información.
- ✓ Aprender la sintaxis básica y los posibles elementos de XML.
- ✓ Diferenciar entre documentos bien formados y documentos válidos.
- ✓ Conocer qué son y para qué se usan los espacios de nombres en XML.

3.1 TIPOS DE LENGUAJES

Dentro de los lenguajes para el almacenamiento y transmisión de la información se pueden encontrar los tipos siguientes:

- **De marcas. XML** (*eXtended Markup Language*). Es un metalenguaje extensible de etiquetas desarrollado por el W3C que permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

```
view plain  copy to clipboard  print  ?
01. <?xml version="1.0" encoding="UTF-8" ?>
02. <poblaciones>
03.   <poblacion id="0">Alcobendas</poblacion>
04.   <poblacion id="1">Miraflores de la Sierra</poblacion>
05.   <poblacion id="2">San Fernando de Henares</poblacion>
06. </poblaciones>
```

Figura 3.1. *Documento XML*

XML es una simplificación y adaptación del SGML.

- **De listas. JSON** (*JavaScript Object Notation*). Es un formato ligero para el intercambio de datos.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX.

AJAX (*Asynchronous JavaScript And XML*, JavaScript Asíncrono y XML) es una técnica de desarrollo web para crear aplicaciones interactivas o **RIA** (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad de las aplicaciones.
Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador semántico de JSON que de XML. En JavaScript, un texto JSON se puede analizar fácilmente usando el procedimiento *eval ()*, lo que ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.
Si bien es frecuente ver JSON posicionado contra XML, también es frecuente el uso de JSON y XML en la misma aplicación. Por ejemplo, una aplicación de cliente que integra datos de *Google Maps* con datos meteorológicos en SOAP hacen necesario soportar ambos formatos.
El término JSON está altamente difundido en los medios de programación, sin embargo, es un término mal descrito ya que en realidad es solo una parte de la definición del estándar **ECMA-262** en que está basado JavaScript. De ahí que ni Yahoo, ni Google emplean JSON, sino **LJS** (*Literal Javascript*). Una de las cualidades intrínsecas de LJS facilita el flujo de datos e, incluso, de funciones. Todo lo referente a transferencia de datos en todos sus tipos no requiere de la función *eval ( )*, y es precisamente en eso en donde supera por mucho JavaScript al XML, si se utiliza el LJS y no la incorrecta definición de JSON.

```
view plain  copy to clipboard  print  ?
01. {"poblaciones":[
02.     {"poblacion": { "@id": "0", "#text": "Alcobendas" }}
03. ,
04.     {"poblacion": { "@id": "1", "#text": "Miraflores de la Sierra" }}
05. ,
06.     {"poblacion": { "@id": "2", "#text": "San Fernando de Henares" }}
07. ]}
```

***Figura 3.2**. Equivalencia del documento XML de la figura 3.1 en JSON*

ACTIVIDADES 3.1

- Vaya a la página *http://www.json.org* y obtenga más información sobre JSON.
- Busque información en la web sobre JSON.
- Busque información en la web sobre AJAX.

3.2 DEFINICIÓN DE XML

XML se puede ver como un subconjunto de SGML mucho más simple, hasta el punto de que la especificación de XML es de una décima parte que SGML. Por otro lado, el lenguaje XML se puede definir como un metalenguaje, esto es, XML puede ser usado para definir otros lenguajes, al igual que el SGML. Por ejemplo, el lenguaje XHTML, ampliamente usado hoy en día en la *web*, se ha construido usando como base XML.

A continuación, se detallará como se construyen los documentos XML y las características principales de los mismos.

3.3 ESTRUCTURA Y SINTAXIS DE XML

Un documento XML está formado, en principio, por lo que se conoce como "texto plano", esto es, texto en el cual todos los caracteres se representan visualmente, sin existir caracteres no visibles exceptuando los de salto de línea, tabulador o espacio.

Tal y como ya sabe el lector los documentos escritos usando XML contendrán marcas para separar la información que estructura el documento de la información que se quiere almacenar. Para construir dichas marcas, en XML se usan los caracteres "<" y ">" para delimitar el texto que se desea marcar, mientras que el carácter "/" sirve para indicar la etiqueta de finalización del marcado. Un posible ejemplo de documento XML sería el siguiente:

<nombre>Luis</nombre>

Esta construcción se denomina habitualmente "elemento" y constituye la base principal de los documentos XML. Además de los elementos, un documento XML puede contener otros tipos de información. A continuación, se especificarán los más relevantes.

3.3.1 ETIQUETAS, ELEMENTOS Y ATRIBUTOS

Las etiquetas son el componente de XML que permite definir los elementos que conformarán un documento de la siguiente forma:

<etiqueta>Valor</etiqueta>

Como se puede observar los elementos se formarán usando una etiqueta de inicio, otra de fin, delimitadas mediante los caracteres ">" y ">", y que comparten el identificador textual pero añadiendo el carácter "/" al principio. En medio de las etiquetas de inicio y fin del elemento se representará el contenido que se desee almacenar en ese elemento. Este contenido puede a su vez englobar otros elementos. Por otro lado, los elementos pueden no contener ningún valor, pero en ese caso se deberá usar solamente la etiqueta de finalización.

Se considera que en el elemento XML engloba todo lo que se encuentra entre las correspondientes etiquetas de inicio y de fin y pueden contener tanto otros elementos como simplemente texto o una combinación de ambos.

A continuación, se muestra un ejemplo de XML para representar la ficha de un alumno sería sintácticamente correcto:

EJEMPLO 3.1

```
<alumno>
    <nombre>Pablo</nombre>
    <apellido>Pérez</apellido>
    <telefono>9155555</telefono>
    <direccion></direccion>
    </varon>
</alumno>
```

Los nombres de los elementos deberán empezar por una letra o bien por el carácter "_" o ":" siempre y cuando el principio no contenga la palabra "xml" en cualquier combinación posible de mayúsculas y minúsculas. Además, los nombres son *case sensitive* y solo podrán contener letras, números y los caracteres "_", ".", "_", ":" Teniendo en cuenta estas reglas, los siguientes ejemplos serían incorrectos:

¿SABÍAS QUE...?

Case sensitive (del inglés, literalmente *sensible a las mayúsculas/minúsculas*) es una expresión usada en jerga informática que se aplica a los textos en los que tiene alguna relevancia escribir un carácter en mayúsculas o minúsculas.

```
<nombre>Pablo</finNombre>
</apellido>Pérez<apellido>
<varon>
<xMl_direccion></xMl_direccion>
<!notas></!notas>
```

Los elementos pueden asimismo contener atributos, los cuales se especificarán en la etiqueta de inicio del elemento. El objetivo de los atributos es poder proporcionar una información adicional sobre un elemento concreto. La sintaxis para representar los atributos consiste en especificar el nombre del atributo dentro de la etiqueta de inicio, a continuación un símbolo "=" y finalmente el valor del atributo delimitado por comillas dobles o por comillas simples:

```
<elem1 atrib1="val1" atrib2="val2" > Valor </elemento>
<elem2 atrib1="val3" atrib3="val3"> Valor </elemento>
```

Siguiendo con el ejemplo presentado previamente, el sexo del alumno anteriormente se había representado usando un elemento mientras que en el siguiente ejemplo se utiliza un atributo para denotar dicha característica. Además se ha añadido el atributo "fechaNacimiento" para el elemento alumno y el atributo "tipo" para el elemento teléfono.

EJEMPLO 3.2

```
<alumno sexo="varon" fechaNacimiento="5/6/1990">
        <nombre>Pablo</nombre>
        <apellido>Pérez</apellido>
        <telefono tipo="movil">9155555</telefono>
        <direccion>Ronda de Segovia 111</direccion>
</alumno>
```

Se puede observar que generalmente se pueden usar tanto atributos como nuevos elementos para representar información. Sin embargo, el uso de un número excesivo de atributos puede provocar que el documento XML sea menos legible, más difícil de mantener y difícilmente extensible. Además hay que tener en cuenta que los atributos no pueden contener información en forma de árbol, esto es, no pueden contener otros elementos o atributos tal y como sucede con los elementos. De forma general, se puede establecer la recomendación no usar atributos en exceso y dejarlos casi exclusivamente para representación de metadatos.

Siguiendo esta recomendación, en el ejemplo tanto el atributo "sexo" como el atributo "fechaNacimiento" se pueden convertir en elementos. Además en este ejemplo se ha añadido un identificador del alumno como atributo:

EJEMPLO 3.3

```
<alumno id="532">
        <nombre>Pablo</nombre>
        <apellido>Pérez</apellido>
        <fechaNacimiento>
                <dia>5</dia>
                <mes>6</mes>
                <año>1990</año>
        </fechaNacimiento>
        <sexo>varón</sexo>
        <telefono tipo="movil">9155555</telefono>
        <direccion>Ronda de Segovia 111</direccion>
</alumno>
```

ACTIVIDADES 3.2

En el ejemplo siguiente:

```
<persona sexo="hombre" id="ricky">
<nombre>Ricky Martin</nombre>
<email>ricky@puerto-rico.com</email>
<relacion amigo-de="leatitia">
</persona>
```

Indique los elementos y los atributos que haya.

3.3.2 CARACTERES ESPECIALES

En XML algunos símbolos son reservados del lenguaje por lo que para poder representarlos es necesario usar unos códigos. Estos se definen usando el símbolo & seguido de una palabra clave y terminados por punto y coma. Estas construcciones son entidades predefinidas (las entidades se verán más adelante). A continuación, se detallan algunos de los más relevantes:

Código	Carácter
&quot;	"
&amp;	&
&apos;	'
&lt;	<
&gt;	>

3.3.3 INSTRUCCIONES DE PROCESAMIENTO

Más allá de los propios datos contenidos en los ficheros XML y las etiquetas de marcado en un fichero XML se pueden encontrar instrucciones especiales llamadas instrucciones de procesamiento. Éstas comienzan con <? Y terminan con ?>. Una de las instrucciones de procesamiento más habituales es la que se usa para indicar que versión de XML se va a usar y cuál es la codificación de caracteres que se va a usar. Por ejemplo, si se usa XML 1.0 y UTF-8 la instrucción de procesamiento sería la siguiente:

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

3.3.4 COMENTARIOS Y SECCIONES CDATA

Dentro de un documento XML se puede añadir información que no pertenezca ni al marcado ni la información contenida documento y que sirve para documentarlo en forma de comentarios internos. La sintaxis de un comentario consta de un texto delimitado por una marca inicial "<!--" y una marca final "-->"

```
<!-- Comentario valido en XML -->
```

Los comentarios son elementos especiales y no necesita ninguna marca de cierre. Además hay que tener en cuenta que dentro de un comentario no se pueden usar dos guiones seguidos "--".

Además en XML se encuentran disponibles las secciones CDATA, que permiten marcar un texto para que éste no sea procesado por el *parser*, es decir, no serán analizadas sintácticamente. CDATA proviene de "*Characther DATA*" (Datos de carácter) en contraposición a datos de marcado. La sintaxis de estas secciones se basa en la etiqueta de inicio "<![CDATA[" y la etiqueta de fin "]]>".

En el siguiente ejemplo las dos definiciones del elemento *4* serían interpretadas de la misma forma:

EJEMPLO 3.4

```
<alumnos>
<![CDATA[
      <alumno id="321">
            <nombre>Luis</nombre>
      </alumno>
]]>
</alumnos>

<alumnos>
      &lt;alumno id"321"&gt;
      &lt;nombre&gt;Luis&lt;/nombre&gt;
      &lt;alumno&gt;
</alumnos>
```

ACTIVIDADES 3.3

- Visite la web oficial de XML: *http://www.w3.org/XML/*.
- Busque por Internet documentos XML que contengan secciones CDATA.

3.4 DOCUMENTOS XML BIEN FORMADOS

Una vez visto los elementos que pueden formar parte de un documento XML y sus características el siguiente paso será establecer cuando un documento es correcto. En este sentido, en XML se puede hablar de documentos "bien formados" y documentos "válidos".

Los documentos bien formados son aquellos que son sintácticamente correctos, es decir, que cumplen las reglas expuestas en los apartados previos. Sin embargo, los documentos válidos son aquellos que, además de estar bien formados, cumplen los requisitos de una definición de estructura.

Las definiciones de estructura se presentarán en el Capítulo 4.

Además de las reglas expuestas en los apartados anteriores, se pueden destacar algunos otros aspectos en los que no se ha incidido de forma directa:

- Un documento XML debe contener un único elemento raíz.
- Los elementos son *case-sensitive*, por lo que las etiquetas de inicio y fin de un elemento deben concordad en mayúsculas y minúsculas.
- El documento solo contendrá caracteres válidos dependiendo del tipo de codificación del documento.
- Los caracteres "<", ">" y "&" solo deben aparecen para delimitar las etiquetas de los elementos y para usar caracteres especiales.

3.5 ESPACIOS DE NOMBRES

De forma general, los documentos XML se suelen combinar con otros documentos XML existentes, esto es, es habitual que se desee usar uno o varios módulos desarrollados previamente por terceras personas. Esta modularidad es una característica esencial de XML y permite al desarrollador poder reutilizar código existente que en muchos casos ya ha sido depurado y ampliamente probado.

El problema que surge en estos casos es la posible colisión que se puede producir en los nombres de los elementos que conformen los módulos que se quieren usar. Para solucionar este problema XML proporciona un mecanismo denominado "espacio de nombres", que permite asignar nombres extendidos a los elementos de forma que se puedan evitar las colisiones.

3.5.1 DECLARACIÓN DE ESPACIOS DE NOMBRES

Un espacio de nombres se define como una referencia URI (*Uniform Resource Identifier*), que servirá para identificar los elementos que pertenecen a dicho espacio de nombres. Otra forma de verlo es que los elementos tendrán un nombre compuesto por dos partes: una primera con su nombre y una segunda con el nombre de espacio de nombres. Este nombre compuesto permitirá identificar de forma unívoca al elemento en cuestión y de esta forma conocer siempre a qué elemento se está refiriendo el documento.

La construcción de estos nombres extendidos se realiza uniendo el nombre el espacio de nombres y el nombre del elemento o atributo usando como conector el símbolo ":". Sin embargo, las referencias URI pueden ser largas, lo

que va en detrimento de la legibilidad y claridad del documento, además de propiciar que se cometan errores más fácilmente. Además las URIs pueden contener caracteres no válidos en XML. Para solucionar este problema, en XML se puede asignar un sinónimo corto al espacio de nombres de forma que este sinónimo corto sea el que se use a lo largo del documento. El sinónimo se asigna usando el separador ":" y la etiqueta "xmlns". En realidad, "xmlns" es un atributo reservado (recuerde que los atributos no pueden comenzar por "xml" en ninguna combinación de mayúsculas y minúsculas). El ejemplo siguiente representa el uso de un espacio de nombres:

EJEMPLO 3.5

```
<elementoej xmlns:enej="http://dominioej.com/rutaej">
    <enej:elemento1>Texto 1</nsej:elemento1>
    <enej:elemento2>Texto 2</nsej:elemento2>
</elementoej>
```

Se puede observar que el sinónimo corto del espacio de nombres de ejemplo es "enej" y que su uso resulta más adecuado que el nombre completo del espacio de nombres "http://dominioej.com/rutaej". Además se puede observar que los elementos "elemento1" y "elemento2" pertenecen al espacio de nombres "enej" al estar calificados con el sinónimo de dicho espacio.

ACTIVIDADES 3.4

- Busque un documento XML por Internet que disponga de un espacio de nombres.

3.5.2 ESPACIOS DE NOMBRES POR DEFECTO

Si un espacio de nombres se declara sin su sinónimo correspondiente esto indicará que todos los elementos (incluido el elemento que declara el espacio de nombres) que contenga pertenecerán a dicho espacio de nombres. Esto será así siempre que los elementos no tengan el prefijo de otro espacio de nombres. Por tanto, sería como definir un espacio de nombres por defecto para los elementos que no tengan espacio de nombres asignado.

Otro uso de los espacios de nombres que puede resultar de gran utilidad es dejar su declaración en blanco (xmlns=""), lo que indicaría que los elementos y atributos contenidos, por defecto, no pertenecen a ningún espacio de nombres.

Por último, hacer énfasis en que cuando de declara un espacio de nombres por defecto (o sin espacio de nombres por defecto, dejando el atributo *xmlns* vacío) pero un elemento contienen un prefijo de un espacio de nombres, el espacio de nombres que prevalecerá será éste último.

RESUMEN DEL CAPÍTULO

En este capítulo se ha presentado una introducción a los lenguajes de marcas así como la motivación de su origen y la evolución que han seguido desde su nacimiento.

Por otro lado, se ha introducido la sintaxis básica de XML y se han definido los elementos básicos que pueden aparecer en un documento XML.

A continuación, se ha expuesto qué es un documento XML bien formado y cuándo es válido, dejando este último punto para su profundización en el Capítulo 4.

Por último, se han introducido el concepto de espacios de nombres y su utilidad dentro de un sistema que maneja múltiples documentos XML.

EJERCICIOS PROPUESTOS

- **1.** Definir un documento XML que permita representar un libro. Deberá contener los atributos típicos como "título", "autores", "editorial", "fecha de publicación", "isbn", etc.
- **2.** A partir de la definición anterior escribir un documento XML que al menos contenga 10 entradas de libros.
- **3.** Buscar un validador de XML *online* (por ejemplo el de W3C *http://validator.w3.org/*), introducir el documento generado en el ejercicio anterior y comprobar que el documento esté bien formado.
- **4.** Crear un espacio de nombres ficticio e introducirlo en el XML del ejercicio 2 y comprobar que el documento XML sigue estando bien formado.