4 TECNOLOGÍAS XML

4.1 XML

4.1.1 INTRODUCCIÓN

4.1.1.1 Qué es XML

El Lenguaje Extensible de Marcas, abreviado XML, describe una clase de objetos de datos llamados documentos XML y define parcialmente el comportamiento de los programas de computadora que los procesan. XML es un "perfil de aplicación" o una forma restringida de SGML, el Lenguaje Estándar Generalizado de Marcas descrito en la norma ISO 8879.

La versión 1.0 del lenguaje XML es una recomendación del W3C desde Febrero de 1998. Se puede considerar a XML como un metalenguaje, es decir, un lenguaje para crear otros lenguajes.

XML, con todas las tecnologías relacionadas, representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Así pues, el XML juega un papel clave en este mundo actual, que tiende a la globalización y la compatibilidad entre los sistemas, ya que es la tecnología que permitirá compartir la información de una manera segura, fiable y fácil. Además, XML permite al programador y los soportes dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse de ello.

XML despliega principalmente su enorme potencial en el mundo de Internet y el de los negocios electrónicos, ya que existen muchos sistemas distintos que tienen que comunicarse entre sí, aunque es perfectamente aplicable a cualquier rama.

4.1.1.2 Objetivos de XML

XML fue diseñado para suplir la carencia de una tecnología similar, con los siguientes objetivos como meta:

- Que fuera idéntico al HTML a la hora de servir, recibir y procesar la información, para aprovechar toda la tecnología implantada para este último.
- Que fuera formal y conciso desde el punto de vista de los datos y la manera de guardarlos.
- Que fuera extensible, para que se pueda utilizar en todos los campos del conocimiento.
- Que fuese fácil de leer y editar, incluso para los humanos.
- Que fuese fácil de implantar, programar y aplicar a los distintos sistemas.

4.1.2 ASPECTOS BÁSICOS

4.1.2.1 Tipos de documentos

Existen dos tipos de documentos XML:

- **Bien formados**: son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas propias de XML, sin estar sujetos a unos elementos fijados en una "gramática". Los documentos XML tienen una estructura jerárquica muy estricta, y los documentos bien formados deben cumplirla.
- **Válidos**: Además de estar bien formados, siguen una estructura y una semántica determinada por una gramática: sus elementos, la estructura jerárquica y los atributos, deben ajustarse a lo que la gramática dicte.

Se deduce fácilmente que todo documento válido es implícitamente bien formado, pero no tiene porqué cumplirse lo contrario.

4.1.2.2 Metalenguajes para definir XML

Para especificar la anteriormente citada "gramática", XML define actualmente dos metalenguajes:

- **Definición de tipos de documento (DTD)**: es una definición de los elementos que puede haber en el documento XML, y las relaciones entre ellos, sus atributos, posibles valores, etc. Cuando se procesa cualquier información formateada mediante XML, lo primero es comprobar si está bien formada, y luego, si incluye referencia a un DTD, se comprueba que sigue sus reglas gramaticales. La DTD puede aparecer de dos maneras:
  - Incluida en el propio documento XML.
  - Como archivo independiente, en formato XML.

  Las DTD usan una sintaxis distinta a XML y, en principio, son muy complicados de crear.
- **Esquemas (Schema XML)**: los Esquemas describen la estructura de la información (al igual que las DTDs). Aparecieron posteriormente para solventar los problemas de las DTD. Un esquema XML es algo similar a un DTD, es decir, define qué elementos puede contener un documento XML, cómo están organizados y qué atributos y de qué tipo pueden tener sus elementos. Están escritos en sintaxis XML, por lo que no hay que aprender un nuevo metalenguaje. Sin embargo, es más complicado realizar una correcta especificación que en DTD.

Un detalle importante de señalar a la hora de hablar de los DTD o XML Schema es que estos lenguajes también permiten comprobar la integridad de los datos en cualquier momento. Se calcula que un 70% de las líneas de código que escribe un programador están orientadas a comprobar la integridad de los datos, es decir, comprobar si donde se supone que hay un número efectivamente lo hay, si el número es entero o cualquier otra comprobación. Los metalenguajes de XML sirven para tomar un documento XML y comprobar que los datos que en él se incluyen son válidos, comprobando si los que tenemos en el XML concuerdan con los que se deberían tener.

Esta operación se puede realizar al leer el documento, liberando de esa carga al programador.

4.1.2.3 Procesado de XML

Los *parsers* son aplicaciones que procesan documentos XML. El procesador XML o procesador de XML es la herramienta principal de cualquier aplicación XML. Aparte del procesado en sí, un procesador XML también comprueba si los documentos están bien formados o válidos. La gran ventaja de usar estos procesadores XML es que se pueden incorporar a las aplicaciones, de manera que éstas manipulen y trabajen con documentos XML directamente, como suele suceder en los navegadores, que incorporan sus propios procesadores XML.

Se puede dividir a los procesadores XML en dos grupos principales:

- **sin validación**: el procesador XML no valida el documento utilizando un DTD o esquema, sino que sólo chequea que el documento esté bien formado de acuerdo a las reglas de sintaxis de XML (sólo hay una etiqueta raíz, las etiquetas están cerradas, etc.).
- **con validación**: además de comprobar que el documento está bien formado según las reglas anteriores, comprueba el documento utilizando un DTD o esquema (ya sea interno o externo).

En el caso de utilizar un DTD, es preferible utilizar un procesador XML con validación.

Existen gran cantidad de procesadores XML, muchos de ellos gratuitos. Entre ellos destacan XP, Xerces, Piccolo, etc.

4.1.3 SINTAXIS BÁSICA

4.1.3.1 Estructura general

A continuación se muestra un ejemplo sencillo de documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<!DOCTYPE mensaje [

   <!ELEMENT mensaje (#PCDATA)>

]>

<!-- este es un comentario -->

<mensaje>Hello, world!</mensaje>
```

Antes de detallar más a fondo la sintaxis, conviene resaltar ciertos aspectos:

- Los documentos XML son sensibles a mayúsculas, esto es, en ellos se diferencian las mayúsculas de las minúsculas.

- Todos los espacios y retornos de carro se tienen en cuenta.
- Los valores de los atributos de todas las etiquetas deben ir siempre entrecomillados. Son válidas las dobles comillas (") y la comilla simple (').

Un documento XML está compuesto de marcas y datos mezclados:

- Las marcas se componen de etiquetas de inicio y de final, entidades, comentarios, delimitadores de CDATA, DTD e instrucciones de procesado.
- Todo lo demás son datos.

En los siguientes apartados, se desarrollarán las marcas.

4.1.3.2 Sintaxis de los elementos

Un documento XML está compuesto por elementos, y en su sintaxis éstos se nombran mediante etiquetas. Cada elemento puede contener a su vez otros elementos o entidades. Las entidades funcionan como abreviaciones. Los elementos pueden contener atributos (propiedades) que indican información sobre como el elemento debe ser procesado.

Hay dos tipos de elementos: los vacíos y los no vacíos. Hay varias consideraciones importantes a tener en cuenta al respecto:

- Toda etiqueta no vacía debe tener una etiqueta de cerrado: `<etiqueta>` debe estar seguida de `</etiqueta>`.
- Todos los elementos deben estar perfectamente anidados: no es válido poner:

```
<ficha><nombre>José Antonio</ficha></nombre>
```

y sí lo es sin embargo:

```
<ficha><nombre>José Antonio</nombre></ficha>
```

- Los elementos vacíos son aquellos que no tienen contenido dentro del documento. La sintaxis correcta para estos elementos implica que la etiqueta tenga siempre esta forma: `<etiqueta/>`.

4.1.3.3 Prólogo y Declaración de un documento

Los documentos en XML se inician con una declaración de XML especificando:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

- **version**: indica la versión de XML usada en el documento. La actual es la versión 1.0. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.
- **encoding**: la forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del procesador XML el entender o no la codificación. Por defecto es UTF-8.

- **standalone**: indica si el documento va acompañado de un DTD ("no"), o no lo necesita ("yes"). En principio no hay porqué ponerlo, porque luego se indica el DTD si se necesita.

  Deben ser seguidos de la declaración de su DTD (document type declaration)

  ```
  <!DOCTYPE mensaje [   <!ELEMENT mensaje (#PCDATA)>    ]>
  ```

  La declaración de su DTD puede hacerse externa al documento:

  ```
  <!DOCTYPE mensaje SYSTEM "mensaje.dtd">
  ```

4.1.3.4 Comentarios

Los comentarios pueden ocurrir en cualquier lugar fuera de una marca:

```
<!-- comentario -->
```

4.1.3.5 Declaraciones de Elementos

Ayuda a restringir el contenido de un elemento. Algunos ejemplos son:

```
<!ELEMENT br         EMTPY>

<!ELEMENT p         (#PCDATA|emph)*>

<!ELEMENT container  ANY>
```

4.1.3.6 Definiciones de Atributos

Los atributos especifican características especiales que aplican a un elemento en particular. Se declaran:

```
<!ATTLIST subject form (bold|italic|normal) "normal">

...

<subject form="italic">...</subject>

<subject>...</subject>
```

4.1.3.7 Entidades

Las entidades sirven para dos propósitos:

- Abreviaturas
- Para integrar caracteres no incluidos en el conjunto de caracteres.

Un ejemplo:

```
<!ENTITY eacute CDATA "&#233;">

<!ENTITY buap "Benem&eacute;rita Universidad Aut&oacute;noma de Sevilla">
```

```
...

&buap;
```

4.1.3.8 Instrucciones de Procesado

No son parte del documento sino que ofrecen al procesador información adicional:

```
<?xml version="1.0"?>
```

4.1.3.9 Secciones CDATA

Permiten integrar texto en un documento en XML que de otra forma sería interpretado como etiquetas:

```
<![CDATA[<poema>This is the sea</poema>]]>
```

4.1.4 TECNOLOGÍAS XML

Debido a la universalidad de XML, se han desarrollado numerosas tecnologías y soluciones utilizando toda la potencia que XML ofrece. Se han creado numerosos lenguajes específicos a partir de XML para suplir la falta de estándares en muchas áreas, o incluso para sustituir los obsoletos y propietarios. El resultado es una nueva concepción de expresar los datos y de enfocar las comunicaciones en la red.

Resultaría imposible describir y mencionar la totalidad de las tecnologías relacionadas con XML. Cada día, se añaden a la lista nuevas soluciones. Se citan a continuación varias tecnologías de distintos campos, para comprobar a manera de ejemplo el alcance y eficacia de XML:

- SMIL: Lenguaje Sincronizado de Integración Multimedia.
- MathML: Lenguaje de Marcas Matemático.
- SVG: Gráficos de Vector Escalables.
- DrawML: Lenguaje de Meta Dibujo Estándares eCommerce.
- cXML: XML para comercio.
- ebXML: XML para negocios electrónicos.
- UBL: Universal Business Language, XML para comercio electrónico.
- DMTF: Manejo Distribuido de Tareas Forzadas.

No se han mencionado tecnologías y productos que usan XML para realizar sus transacciones o autodefinirse, como puede ser JXTA, WSDL, etc., porque la lista sería interminable.

4.2 XML Y JAVA

Las aplicaciones actuales, como servicios web, aplicaciones web, etc. dependen de la capacidad de los participantes de poder comunicarse aunque usen distintos sistemas de información. XML es una tecnología clave para resolver este problema, puesto que define una nueva manera de interoperabilidad de información entre sistemas muy dispares.

Así mismo, las aplicaciones también dependen de la capacidad de interoperabilidad entre plataformas completamente distintas. La solución aquí es Java, ya que es independiente de la plataforma y posee código portable.

Por estas dos razones, es evidente que Java y XML juntos constituyen una poderosa herramienta de desarrollo, obteniendo la independencia de plataforma y de sistema. De esta manera, han surgido recientemente multitud de APIs para el tratamiento de XML. Entre las opciones más destacables están:

- **JAXP** (*Java API for XML Processing*): es la API referencia para el manejo básico de XML proporcionada por Sun Microsystems. Aglutina a 3 APIs:
    - SAX: para procesado general de XML.
    - DOM: para procesado bajo la especificación DOM de documentos XML.
    - XSLT: para realizar las transformaciones XSLT.

  Es muy completa, y con ella se puede hacer prácticamente de todo.
- **JAXB** (*Java Architecture for XML Binding*): genera clases Java a partir de esquemas XML. La proporciona Sun.
- **JAXR** (*Java API for XML Registries*): proporciona formas de acceder a registros de negocios en Internet. También está proporcionada por Sun.
- **JAX-RPC** (*Java API for XML-Based RPC*): implementación de RPC basado en SOAP. También está proporcionada por Sun.
- **JAXM** (*Java API for XML Messaging*): interfaz para el intercambio de mensajes y establecimiento de transacciones de negocios. Está basado en SOAP.
- **dom4j**: es un marco de trabajo para el manejo de XML basado en DOM y SAX. Es un proyecto de software libre.
- **JDOM**: librerías para manipular DOM optimizado para Java.
- **Xerces**: procesador XML, está muy extendido su uso. Pertenece al Apache XML Project.
- **Xalan**: procesador de hojas de estilo XSLT, también pertenece al Apache XML Project.

Se ha citado aquí sólo una pequeñísima parte del conjunto de software desarrollado para el manejo de XML en Java, aunque son generalmente los más usados.

Página intencionadamente dejada en blanco.