5.- XML

5.1.-Necesidad de XML

Una vez establecidos los tipos de datos que se van a intercambiar, es fundamental encontrar un mecanismo para que dichos datos sean serializados y se posibilite su intercambio entre diferentes tecnologías a través de la red. Para ello desde hace unos años se viene imponiendo el uso de XML (eXtensible Markup Language). XML es un metalenguaje, esto es, un lenguaje para crear y describir otros lenguajes, lo que lo convierte en una herramienta tremendamente potente y flexible, ya que pueden crearse lenguajes y etiquetas específicos según las necesidades. Aunque, naturalmente, para que múltiples aplicaciones usen el mismo documento XML, tienen que estar de acuerdo en los nombres de etiquetas que intentan usar.
Otra de sus fortalezas es la separación del contenido y presentación, lo que permite principalmente:

- Independencia de plataforma: posibilitando acceder a la misma información en un PC, un teléfono móvil, un PocketPC, etc. con presentaciones adecuadas a cada entorno.
- Elevado nivel de automatización: al dar las etiquetas información sobre el contenido de las mismas y no sobre su presentación, las aplicaciones pueden realizar, de manera automática, tareas mucho más complejas que hasta ahora.

5.2.-¿Por qué es importante el XML?

Hay un gran número de razones para aceptar el uso de XML. Esta sección lista algunos de los más importantes.

Texto Plano.
Como XML no está en formato binario, podemos crear y editar ficheros con cualquiera de los editores de texto estándar. Esto es útil para el almacenamiento de pequeñas cantidades de datos. Por otro lado, el XML con bases de datos hace posible almacenar grandes cantidades de datos de forma eficiente. Por eso XML proporciona escalabilidad para cualquier cosa, desde pequeños ficheros de configuración a repositorios de datos de grandes compañías.

Identificación de Datos.
XML dice qué clase de datos tenemos, no cómo se muestran. Como las etiquetas de marca identifican la información y dividen los datos en partes, éstas pueden ser usadas de diferentes formas por diferentes aplicaciones.

Formato de presentación.
Cuando el estilo es importante, la hoja de estilos estándar, XSL, nos dicta cómo mostrar los datos. Por supuesto, podríamos haber hecho lo mismo con HTML, pero no podríamos procesar los datos con programas de búsqueda y programas de extracción. Más importante, como XML está libre de estilo podemos usar una hoja de estilo completamente diferente para producir salida en postcript, TEXT, PDF, o algún nuevo formato que no haya sido inventado todavía.

Reutilización en Línea.
Uno de los aspectos más prácticos de los documentos XML es que pueden estar compuestos por entidades separadas. Podemos hacer esto con HTML, pero sólo

enlazando a otros documentos. Al contrario que HTML, las entidades XML pueden ser incluidas "en línea" en un documento. Las secciones incluidas parecen una parte normal del documento -- podemos buscar en el documento completo a la vez o descargarlo en una sola pieza. Esto nos permite modularizar nuestros documentos sin recurrir a los enlaces.

Fácilmente Procesable.
Como se mencionó antes, la notación normalizada y consistente hace fácil construir un documento y procesar datos XML. En XML, la etiqueta **<dt>** siempre debe tener un terminador **</dt>**, o será definida como una etiqueta **<dt/>**. Esta restricción es una parte crítica de las restricciones que crean un documento XML bien-formateado. (De otra forma, el analizador XML no podrá leer los datos). Esta normalización en la nomenclatura hace que un documento XML sea fácilmente procesable.

Herencia.
Finalmente, los documentos XML se benefician de su estructura en forma de árbol. Estas estructuras son, en general, rápidas de acceder porque podemos trasladar la parte que necesitemos, como pasando a través de una tabla de contenidos.

5.3.-Realización de la transformación Java-XML

Existen multitud de APIS de java que te permiten trabajar con documentos de XML, con diferentes funcionalidades. Nuestra principal finalidad es transformar documentos XML a objetos Java y viceversa. En el mundo Java la forma básica de abordar este problema es usar **JAXP** para parsear el documento (usando SAX o DOM según convenga) y generar a mano los objetos, dando valor a sus atributos; en cualquier caso esta es una labor tediosa. En la actualidad contamos con una serie de herramientas de más alto nivel que nos permiten realizar esta tarea de una forma cómoda. Entre ellas podemos nombrar:

- Java Architecture for XML Binding (JAXB): Esta es la solución estándar de SUN para la extracción de objetos Java a partir de entidades XML y viceversa.
- XStream: librería simple que permite pasar de documentos XML a objetos Java con una mínima configuración. Adicionalmente permite realizar el proceso contrario, es decir, serializar objetos Java en formato XML. Estas características lo hacen útil en escenarios en los que es necesario que la información sea persistente sin recurrir a soluciones más complejas como sistemas de bases de datos.
- Castor: Es un framework (Java y open source) para acceso a datos, que permite mapear objetos a documentos XML o bases de datos relacionales. Tiene una funcionalidad muy amplia.
- XMLBeans: Este proyecto de apache todavía está en la incubadora pero promete convertirse en una herramienta imprescindible para el tratamiento de XML. La idea es aprovecharse de la riqueza y las características de XML y del XML Schema traduciéndolas de la manera más natural posible a sus equivalentes Java y a las construcciones típicas del lenguaje.
- Digester: se inició como una parte de Jakarta Struts (un framework para el desarrollo de aplicaciones web J2EE). Su propósito inicial fue procesar el fichero de configuración de una aplicación web basada en Struts. Posteriormente, dada su evidente utilidad para una gran heterogeneidad de aplicaciones, fue incluido como un framework independiente dentro del proyecto commons (un repositorio de componentes Java reutilizables). Nos oculta el proceso de parseo con JAXP y nos permite pasar directamente del XML a colecciones de objetos.

- Skaringa: API para enlazar Java y lenguaje XML. Transforma objetos Java a documentos XML y viceversa, y puede generar definiciones de schema XML para una clase Java. Aplicaciones típicas son intercambio de datos, persistencia de objetos, transformación de objetos...

Después de considerar ventajas y desventajas de las diferentes herramientas, se ha optado por usar Castor, por su gran funcionalidad y amplia implantación. También es importante destacar que permite establecer los nombres de las etiquetas específicamente, y el uso de Schemas en caso de que sea necesario.