JSON

1.2 Lenguajes de listas. JSON

En vez de etiquetar la información, los lenguajes de listas la organizan en listas de elementos. JSON (*JavaScript Object Notation* – notación de objetos de Javascript) es un estándar abierto que hereda la sintaxis del lenguaje JavaScript. Se suele utilizar en combinación con AJAX, una tecnología de desarrollo web en alza.

XML

El metalenguaje XML (Lenguaje de Marcado Extensible, por sus siglas en inglés) es un conjunto de reglas para codificar documentos de manera que sean legibles tanto para humanos como para máquinas. Se utiliza principalmente para estructurar, almacenar y transportar datos de manera jerárquica y semiestructurada.

Metalenguaje: lenguaje utilizado para describir un sistema de lenguaje de programación.

XML son las siglas de (*eXtensible Markup Language* – lenguaje extensible de marcas). Se trata de un estándar del W3C cuyo objetivo original consistía en permitir afrontar los retos de la publicación electrónica de documentos a gran escala. Actualmente XML está desempeñando un papel muy importante en el intercambio de una gran variedad de información en la web y en otros contextos.

XML es un subconjunto del lenguaje de marcas SGML (estándar ISO-8879). Su diseño busca la simplicidad de implementación y la interoperabilidad con SGML y HTML, así como su uso por parte de aplicaciones centradas en los datos. Nació en 1996 con los siguientes objetivos, tal como se enumeran en el documento de definición del estándar:

1. XML debe ser utilizable directamente sobre Internet.
2. XML debe soportar una amplia variedad de aplicaciones.
3. XML debe ser compatible con SGML.
4. Ha de resultar fácil escribir programas que procesen documentos XML.
5. El número de características opcionales en XML debe ser mínimo, idealmente cero.
6. Los documentos XML deben ser legibles por un ser humano y razonablemente claros.
7. El diseño de XML se debe preparar rápidamente.
8. El diseño de XML debe ser formal y conciso.
9. Los documentos XML han de ser fáciles de crear.
10. La brevedad en la marcación reviste mínima importancia.

XML se creo por las carencias de HTML

XML, por contra, proporciona:

- Extensibilidad: se pueden definir nuevas marcas y atributos según sea necesario.
- Estructura: se puede modelizar cualquier tipo de datos organizado jerárquicamente.
- Validación: podemos validar automáticamente los datos (de forma estructural).
- Independencia del medio: podemos publicar el mismo contenido en multitud de medios.

SGML

Básicamente, el **SGML** (Standard Generalized Markup Language) es un modelo internacional que permite estandarizar y definir formalmente un documento.

2.2 Características de XML

Un objeto XML (o un documento XML) se define como un documento formado por etiquetas y valores que cumple con la especificación de XML, y que estará bien formado. Cada etiqueta se llama "elemento" y, además de contener un valor literal, puede contar con uno o varios valores auxiliares asignados a identificadores llamados "atributos". En el siguiente fragmento de código XML el elemento Cantidad toma el valor "3", y cuenta con un atributo unidad cuyo valor es "pieza".

```
<Cantidad unidad="pieza">3</Cantidad>
```

Un elemento XML debe tener una etiqueta de cierre, o bien ser un elemento vacío. En el siguiente ejemplo el elemento <Nombre> tiene su correspondiente etiqueta de cierre </Nombre>, mientras <Foto> es un elemento vacío. No contiene valor, pero presenta una barra de cierre (<Foto />).

Diseño de XML: elección de elementos o atributos

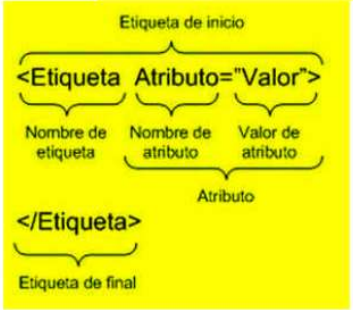
A la hora de decidir qué ítem de información almacenará sus valores en un elemento y cuál en un atributo, se pueden seguir los siguientes criterios:

- Los elementos representan jerarquías.
  - El orden en que aparecen importa.
  - Puede haber múltiples ocurrencias de un elemento, pero no de un atributo.
- Los atributos van asociados a los elementos.
  - Suelen representar metadatos, información auxiliar o de control
  - El orden en que aparecen no importa.

Esto es un documento XML. Todos los documentos XML bien formados deben iniciarse con una instrucción de procesamiento que indique la versión de XML a utilizar:

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

Este es el formato tipo de una etiqueta XML:



En XML, un documento bien formado es el que cumple las siguientes normas:

- Debe haber un único elemento raíz que tendrá como descendientes a todos los demás elementos.
- Todas las etiquetas deben estar cerradas. En HTML podemos trabajar con un gran nivel de relajación de

correspondiente etiqueta de cierre. Esto se debe a que las etiquetas en XML representan una información jerárquica que nos indica cómo se relacionan los diferentes elementos entre sí. Si no cerramos las etiquetas, introducimos en esta representación una serie de ambigüedades que nos impiden un procesamiento automático.

- No puede haber solapamiento de etiquetas. Una etiqueta que se abre dentro de otra etiqueta debe cerrarse antes de cerrar la etiqueta contenedora. Por ejemplo:

- Los valores de los atributos deben estar siempre entrecomillados (a diferencia de HTML, donde podemos declarar atributos sin comillas). Son válidas tanto las comillas simples (') como las dobles ("), pero nunca mezcladas en el mismo valor de atributo.
- Los caracteres <, > y " deben representarse siempre mediante entidades de carácter (usando `<`, `>` y `&quot;`). Estos caracteres son especiales para XML.

La importancia de que un documento esté o no bien formado en XML deriva de que un documento bien formado puede ser analizable sintácticamente. Existen multitud de *parsers* (analizadores) en numerosos lenguajes de programación que nos permiten trabajar con los documentos XML. Los *parsers* de XML son capaces de detectar los errores estructurales de los documentos XML (es decir, si están o no bien formados) y notificárselo al programa. Esta funcionalidad es importantísima para un programador, ya que le libera de la tarea de detectar errores para encomendársela a un programa (el *parser*) que lo hará por sí solo.

Algunos *parsers* van más allá de la simple capacidad de detectar si el documento está bien formado o no, siendo capaces de juzgar la validez del documento, entendiendo por válido el hecho de que la estructura,

posición y número de etiquetas sean correctos y tengan sentido. Imaginemos por un momento el siguiente pedazo de nuestro documento de recetas:

ESPACIOS DE NOMBRES

Evidentemente en este caso tendríamos un problema, ya que no podríamos distinguir el significado de <direccion> en cada uno de los casos. Para ello, en 1999 el W3C definió una extensión de XML llamada espacios de nombres (*namespaces*) que permite resolver conflictos y ambigüedades de este tipo. Como consisten en una colección de nombres y definiciones también se les llama vocabularios.

Los espacios de nombres son un prefijo a las etiquetas de XML para indicar a qué contexto se refiere la etiqueta en cuestión. En el ejemplo anterior podríamos definir:

Para usar un espacio de nombres en un documento, debemos declararlo previamente. Esta declaración puede tener lugar en el elemento raíz del documento de la siguiente forma:

La definición consta de unos atributos `xmlns` (XML *namespace*) donde proporcionamos el prefijo que usaremos para el espacio de nombres y una URI (*Uniform Resource Identifier* – identificador uniforme de recurso) que será un identificador único del espacio de nombres, aunque realmente no se comprueba mediante conexión alguna.

Espacio de nombres por defecto

A la hora de asignar un documento existente a un espacio de nombres, prefijar todas las etiquetas es muy tedioso. Por ese motivo se puede definir un espacio de nombres sin prefijo que afectará al elemento donde se declare y a todos sus descendientes. Es el llamado espacio de nombres por defecto.

El uso del espacio de nombres por defecto presenta un problema: solo afecta a elementos, no a atributos. Por ese motivo en el ejemplo anterior el atributo unidad carecería de espacio de nombres.

Se puede desasignar a un elemento de un espacio de nombres por defecto con la orden:

```
<nombre_elemento xmlns="">
```

DTD

W3C ha desarrollado algunos estándares de XML que nos permiten definir esquemas y vocabularios en documentos XML. Dichos estándares son DTD y XSchema.

DTD es un estándar antiguo, derivado de SGML y que adolece de algunas deficiencias graves, siendo la más grave de ellas el hecho de no estar escrito en XML. XSchema, por otro lado, es un estándar relativamente moderno, muy potente y extensible, que además está escrito íntegramente en XML.

3.1 DTD

DTD (*Document Type Definition* – definición de tipo de documento) es un estándar que nos permite definir una gramática que deben cumplir nuestros documentos XML para ser válidos. Una definición DTD para un documento XML especifica qué elementos pueden existir en el documento, qué atributos pueden tener, qué elementos pueden o deben aparecer contenidos en otros elementos y en qué orden.

Los *parsers* de XML que son capaces de validar documentos con DTD leen esos documentos y el DTD asociado. En caso de que el documento XML no cumpla los requerimientos que le impone el DTD, nos advertirán del error y no validarán el documento. Mediante los DTD definimos cómo será nuestro dialecto de XML (ya que nosotros definimos qué etiquetas vamos a usar en nuestros documentos, qué significado les

A pesar de que DTD es un estándar que deberá ser sustituido por XML Schema, sigue siendo muy usado. Su uso resulta más simple que el de XML Schema y es más compacto. A eso hay que añadir que las mejoras que aporta XML Schema no son necesarias para muchos ficheros XML. Con DTD se han definido multitud de

El valor por defecto puede ser uno de los siguientes:

Valor	Descripción
valor	El valor por defecto del atributo
#REQUIRED	El valor del atributo debe aparecer obligatoriamente en el elemento
#IMPLIED	El atributo no tiene por qué ser incluido
#FIXED valor	El valor del atributo es fijo

Símbolo	Descripción
()	Los paréntesis agrupan subetiquetas `<!ELEMENT Ingrediente (Cantidad,Item)>`

,	Ordenación exacta de los elementos
	`(Nombre, Descripcion?,`
	`Ingredientes?, Instrucciones?)`

|	Uno sólo de los elementos indicados
	`(Cocer | Freir)`
	Si no indicamos nada los elementos aparecen una sola vez
	`(Cantidad, Item)`

?	Elemento opcional `Instrucciones?`
+	Una o más veces `Paso+`
*	Cero o más veces `Ingrediente*`
#PCDATA	*Parsed Character Data* `<!ELEMENT Item (#PCDATA)>`