Lenguajes de Marcas y Sistemas de Gestión de información

Unidad Didáctica 01 - Introducción a los lenguajes de marcas

Francisco Jesús Delgado Almirón

Ciclo formativo de grado superior
Desarrollo de aplicaciones web

CONTENIDO

1.- LENGUAJES DE MARCAS

En el contexto de la programación web, el lenguaje de marca se refiere a la forma de codificar un documento o página web. Es una manera de definir la estructura del texto o su presentación incorporando etiquetas o marcas que contienen información adicional. Su principal objetivo es clarificar la estructura de un documento y el contenido semántico del mismo.

Muchas personas confunden los lenguajes de marcado, como también se les conoce, con los lenguajes de programación, pero no son lo mismo. El lenguaje de marca, por ejemplo, no tiene funciones aritméticas o variables, como sí incluyen los lenguajes de programación.

De hecho, el lenguaje de marcado posee algunas características distintivas, como el uso de texto plano, de forma que cualquier persona puede leer y editar la información. Es una peculiaridad muy útil porque permite realizar modificaciones desde un editor de texto, sin necesidad de recurrir a un software específico para ello.

Además, es fácil acceder a su contenido a través de cualquier dispositivo y altamente flexible, ya que permite combinar con otros lenguajes. También se trata de un lenguaje compacto en el que las etiquetas de marcas se unen con el contenido del mismo.

Un ejemplo de Lenguaje de Marcas sería el siguiente, en el que vemos cómo se encierra el texto entre etiquetas para darle un significado o una representación:

undefined - ejemplo.xml

```xml
<carta>
    <fecha>22/11/2006</fecha>
    <presentacion>Estimado cliente:</presentacion>
    <contenido>bla bla bla bla …</contenido>
    <firma>Don José Gutiérrez González</firma>
</carta>
```

Código 1: Ejemplo de lenguaje de marcas

1.1.- EVOLUCIÓN DE LOS LENGUAJES DE MARCAS

El lenguaje de marcado ha ido evolucionando a lo largo del tiempo:

SGML (*Standard Generalized Markup Language*)

Se trata de un lenguaje de marca que se utilizó aproximadamente hasta finales de 1980, cuando se excluyó del estándar ISO 8879. Cayó en desuso, fundamentalmente, por su dificultad y el hecho de que demandaba herramientas de software demasiado costosas. En la actualidad, ya no se utiliza.

Lenguaje de marcado de hipertexto (HTML)

Es el lenguaje de la web, de manera que la inmensa mayoría de las páginas que existen están escritas en HTML. En 1991 la situación cambió drásticamente cuando Tim Berners-Lee, que conocía el SGML, utilizó su sintaxis para crear el HTML y compartir información entre científicos. Surgió de la necesidad de organizar, enlazar y compatibilizar la información proveniente de diferentes sistemas. Así se unieron dos estándares existentes: ASCII como codificador de caracteres y SGML para dar estructura al texto.

Básicamente, el *Hyper Text Markup Language* define los contenidos de un sitio web de forma textual y estructurada indicando al navegador cómo debe visualizarse el sitio. Su rápida expansión se debió al hecho de que era muy fácil de entender, lo que hizo que se convirtiera en un estándar general para el desarrollo de sitios y páginas web.

HTML5

Con la llegada de HTML5, no solo se incorporan nuevas etiquetas que agregan significado a la página, sino que también se puede añadir audio y vídeo sin recurrir a usar Flash u otro reproductor multimedia.

Otra gran ventaja de desarrollar aplicaciones HTML5 es que el resultado final es completamente accesible desde un ordenador, tableta o móvil. Los navegadores modernos más populares soportan HTML5, por lo que los usuarios pueden visualizar el contenido correctamente. Por supuesto, todo ello facilita y agiliza el proceso de programación, por lo que conocer este lenguaje de marcado es fundamental.

Lenguaje de marcado extensible (XML)

El lenguaje marcado extensible (XML) desempeñó un papel crucial en el éxito de la World Wide Web Consortium en 1998 y seguirá siendo una capa fundacional en la Web 3.0.

Tras varios años creando diferentes especificaciones, a mediados del 2000, se creó la normativa ISO que definió HTML 4.01 (strict) como estándar internacional XML, un lenguaje de marca estructural que no tiene información sobre el diseño.

Luego aparecieron alternativas basadas en XML y en 2004 se creó el Web Hypertext Application Technology Working Group para dar vida a un nuevo estándar e intercambiar datos en la web.

1.2.- ETIQUETAS

Los lenguajes de marcas utilizan una serie de etiquetas especiales intercaladas en un documento de texto sin formato. Dichas etiquetas serán posteriormente interpretadas por los intérpretes del lenguaje y ayudan al procesado del documento.

Las etiquetas se escriben encerradas entre ángulos, es decir < y >. Normalmente, se utilizan dos etiquetas: una de inicio y otra de fin para indicar que ha terminado el efecto que queríamos presentar. La única diferencia entre ambas es que la de cierre lleva una barra inclinada "/" antes del código.

<etiqueta>texto que sufrirá las consecuencias de la etiqueta</etiqueta>

1.3.- VENTAJAS PARA EL TRATAMIENTO DE LA INFORMACIÓN

Los lenguajes de marcas permiten organizar la información, explicar qué tipo de información estamos almacenando, en qué unidades o formato está, dar formato o énfasis a partes o toda la información, dar instrucciones para procesarla, mostrarla al usuario final, etc.

Si la información no está estructurada, es decir, está en texto plano, para saber un dato concreto, tendríamos que extraer dicho dato, aplicando técnicas de lo que se conoce como "Data Mining" o minería de datos. En muchas ocasiones esto implica usar incluso técnicas de inteligencia artificial.

Además, cuando hablamos de información, no hablamos sólo de texto, también de gráficas, imágenes, etc.

2.- CLASIFICACIÓN DE LOS LENGUAJES DE MARCAS

Los Lenguajes de Marcas se pueden clasificar de la siguiente forma:

- **Lenguajes de presentación:** Define el formato (apariencia) del texto. Éstos suelen ocultar las etiquetas y mostrar al usuario solamente el texto con su formato.
- **Lenguajes de procedimientos:** Orientado también a la presentación, pero, además, el programa que representa el documento debe interpretar las etiquetas para realizar acciones en función de ellas.
- **Lenguajes descriptivos o semánticos:** Describen las diferentes partes en las que se estructura el documento, es decir, definen su contenido, pero sin especificar cómo deben representarse.

3.- EJEMPLO DE LENGUAJE DE MARCAS: XML

XML, siglas en inglés de *e**X**tensible **M**arkup **L**anguage*, traducido como 'Lenguaje de Marcado Extensible' o 'Lenguaje de Marcas Extensible', es un metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

XML no ha nacido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

3.1.-VENTAJAS DEL XML

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos PostgreSQL y comunicarla con otra aplicación en Windows y base de datos MS-SQL Server.
- Transformamos datos en información, pues se les añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

3.2.- ESTRUCTURA DE UN DOCUMENTO XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de este como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

A continuación, se muestra un ejemplo para entender la estructura de un documento XML:

undefined - Ejemplo Estructura XML.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
     <Mensaje>
          <Remitente>
               <Nombre>Nombre del remitente</Nombre>
               <Mail> Correo del remitente </Mail>
          </Remitente>
          <Destinatario>
               <Nombre>Nombre del destinatario</Nombre>
               <Mail>Correo del destinatario</Mail>
          </Destinatario>
          <Texto>
               <Asunto>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Asunto>
               <Parrafo>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Parrafo>
          </Texto>
     </Mensaje>
</Edit_Mensaje>
```

Código 2: Ejemplo estructura XML

Aquí está el ejemplo de código del DTD del documento «Edit_Mensaje.dtd»:

undefined - Edit_Mensaje.dtd

```dtd
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- Este es el DTD de Edit_Mensaje -->

<!ELEMENT Mensaje (Remitente, Destinatario, Texto)*>
<!ELEMENT Remitente (Nombre, Mail)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Mail (#PCDATA)>
<!ELEMENT Destinatario (Nombre, Mail)>
<!ELEMENT Nombre (#PCDATA)>
<!ELEMENT Mail (#PCDATA)>
<!ELEMENT Texto (Asunto, Parrafo)>
<!ELEMENT Asunto (#PCDATA)>
<!ELEMENT Parrafo (#PCDATA)>
```

Código 3: Fichero Edit_Mensaje.dtd

3.3.- DOCUMENTOS XML BIEN FORMADOS Y CONTROL DE ERRORES

Los documentos denominados como «bien formados» (del inglés *well formed*) son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, analizarse correctamente por cualquier analizador sintáctico (parser) que cumpla con la norma. Se separa esto del concepto de validez que se explica más adelante.

- Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.
- Los documentos XML solamente permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial.
- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.
- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.
- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.
- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos «entendibles» por las personas.

3.4.- PARTES DE UN DOCUMENTO XML

Un documento XML está formado por el prólogo y por el cuerpo del documento, así como texto de etiquetas que contiene una gran variedad de efectos positivos o negativos en la referencia opcional a la que se refiere el documento, hay que tener mucho cuidado de esa parte de la gramática léxica para que se componga de manera uniforme.

Prólogo

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

El prólogo de un documento XML contiene:

- Una declaración XML. Es la sentencia que declara al documento como un documento XML.
- Una declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
- Uno o más comentarios e instrucciones de procesamiento.

Ejemplo:

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

Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener solo un elemento raíz, característica indispensable también para que el documento esté bien formado. Sin embargo, es necesaria la adquisición de datos para su buen funcionamiento.

Ejemplo:

```
<Edit_Mensaje>
    (...)
</Edit_Mensaje>
```

Elementos

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

Atributos

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.

Por ejemplo, un elemento «estudiante» puede tener un atributo «Mario» y un atributo «tipo», con valores «come croquetas» y «talento» respectivamente.

```
<Estudiante Mario="come croquetas" tipo="talento">Esto es un día que Mario va paseando…</Estudiante>
```

Entidades predefinidas

Entidades para representar caracteres especiales para que, de esta forma, no sean interpretados como marcado en el procesador XML.

Ejemplo: entidad predefinida: & carácter: &.

Secciones CDATA

Es una construcción en XML para especificar datos utilizando cualquier carácter sin que se interprete como marcado XML. No confundir con 2(#PCDATA) que es para los elementos. Permite que caracteres especiales no rompan la estructura. Ejemplo:

```
<![CDATA[contenido especial: \n áéíóúñ&]]>
```

Comentarios

Comentarios a modo informativo para el programador que han de ser ignorados por el procesador. Los comentarios en XML tienen el siguiente formato:

```
<!--- Esto es un comentario --->
<!-- Otro comentario -->
```

3.5.- VALIDACIÓN DE UN DOCUMENTO XML

Que un documento esté «bien formado» solamente se refiere a su estructura sintáctica básica, es decir, que se componga de elementos, atributos y comentarios como XML especifica que se escriban. Ahora bien, cada aplicación de XML, es decir, cada lenguaje definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento.

Esta relación entre elementos se especifica en un documento externo o definición (expresada como DTD o como XSchema). Crear una definición equivale a crear un nuevo lenguaje de marcado, para una aplicación específica.

Document Type Definition

La *Document Type Definition* o DTD (en español "definición de tipo de documento") define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD son denominados válidos.

Declaraciones tipo elemento

Los elementos deben ajustarse a un tipo de documento declarado en una DTD para que el documento sea considerado como válido.

Modelos de contenido

Un modelo de contenido es un patrón que establece los subelementos aceptados, y el orden en que se aceptan.

Declaraciones de lista de atributos

Los atributos se usan para añadir información adicional a los elementos de un documento.

Existen los siguientes tipos de atributos:

- Atributos CDATA y NMTOKEN.
- Atributos enumerados y notaciones.
- Atributos ID e IDREF.

Declaración de entidades

XML hace referencia a objetos que no deben ser analizados sintácticamente según las reglas XML, mediante el uso de entidades. Las entidades pueden ser:

- Internas o externas.
- Analizadas o no analizadas.
- Generales o parametrizadas.

XML Schemas (XSD)

Un Schema es algo similar a un DTD. Define qué elementos puede contener un documento XML, cómo están organizados y qué atributos y de qué tipo pueden tener sus elementos.

El uso de los XSD frente a los DTD presenta las siguientes ventajas:

- Usan sintaxis de XML, al contrario que los DTD.
- Permiten especificar los tipos de datos.
- Son extensibles.

3.6.- ESPACIOS DE NOMBRE EN XML

Los espacios de nombre en XML (*Namespaces*) son una forma de evitar conflictos de nombres cuando se utilizan múltiples vocabularios XML en un solo documento. Esencialmente, permiten distinguir entre diferentes elementos o atributos que podrían tener el mismo nombre pero provenir de diferentes contextos o dominios.

Un espacio de nombre es un identificador que se asocia con un prefijo, de manera que se puede diferenciar a qué dominio pertenece un determinado elemento o atributo. El espacio de nombre se define mediante una URI (*Uniform Resource Identifier*), aunque no tiene por qué apuntar a una página web. Es simplemente un identificador único.

Ventajas de los espacios de nombre en XML:

- **Evitan conflictos de nombres:** Los espacios de nombre resuelven el problema de tener diferentes elementos con el mismo nombre en un mismo documento XML. Esto es especialmente útil cuando se combinan documentos de diferentes orígenes o cuando se integran múltiples vocabularios XML.
- **Mejor integración:** Permiten la integración de diferentes vocabularios XML en un solo documento, como puede ocurrir en aplicaciones complejas que utilizan múltiples estándares o formatos de datos.
- **Claridad y desambiguación:** Facilitan la claridad en los documentos XML, dejando explícito de dónde proviene cada elemento o atributo y su significado dentro de su contexto.
- **Modularidad:** Ayudan a mantener módulos XML independientes, lo cual facilita la reutilización de estos módulos en distintos contextos.

Ejemplo de uso de espacios de nombre en XML

Imaginemos que estamos utilizando dos vocabularios XML diferentes: uno para representar información de libros y otro para representar datos de revistas. Ambos vocabularios tienen un elemento *title*. Para evitar conflictos, usamos espacios de nombre.

undefined - Ejemplo espacios de nombre.xml

```xml
<?xml version="1.0"?>
<catalog xmlns:book="http://www.ejemplo.org/libros"
         xmlns:mag="http://www.ejemplo.org/revistas">
    <book:book>
        <book:title>XML para principiantes</book:title>
        <book:author>Juan Pérez</book:author>
    </book:book>
    <mag:magazine>
        <mag:title>Revista de Tecnología</mag:title>
        <mag:editor>Lucía Gómez</mag:editor>
    </mag:magazine>
</catalog>
```

Código 4: Ejemplo espacios de nombre

Explicación del ejemplo:

En el elemento raíz <catalog>, se definen dos espacios de nombre usando los atributos xmlns:book y xmlns:mag.

- xmlns:book="http://www.ejemplo.org/libros" es el espacio de nombre para los elementos relacionados con libros.
- xmlns:mag="http://www.ejemplo.org/revistas" es el espacio de nombre para los elementos relacionados con revistas.

Dentro del documento XML, los elementos prefijados con book: pertenecen al espacio de nombre definido por http://www.ejemplo.org/libros, y los prefijados con mag: pertenecen al espacio de nombre definido por http://www.ejemplo.org/revistas.

Este mecanismo permite que el elemento title pueda ser utilizado en ambos contextos (book:title y mag:title) sin causar ambigüedades. Esto facilita la combinación de diferentes tipos de datos dentro de un mismo documento XML.

BIBLIOGRAFÍA

Morales, R. (2018, November 25). Qué son los lenguajes de marcas. Ticarte.com. https://www.ticarte.com/contenido/que-son-los-lenguajes-de-marcas

¿Qué es el lenguaje de marca o lenguaje de marcado? (2023, January 24). Universidad Europea. https://universidadeuropea.com/blog/que-es-lenguaje-marca/

Ventajas para el tratamiento de la información. (n.d.). Github.io. Retrieved September 7, 2024, from http://juangualberto.github.io/lmsgi/tema01/ventajas_para_el_tratamiento_de_la_informacin.html

Wikipedia contributors. (n.d.-a). Extensible Markup Language. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Extensible_Markup_Language&oldid=160549383

Wikipedia contributors. (n.d.-b). Lenguaje de marcado. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Lenguaje_de_marcado&oldid=161565642

OpenAI. (2024). ChatGPT (versión GPT-4). https://chat.openai.com/