INTRODUCCIÓN

El documento que a continuación se presenta es un estudio teórico sobre protocolos, acompañado de ejemplos sencillos que sirven de guía para el alumno.

El documento se estructura de la siguiente manera:

En el **Capítulo 1** se realiza una introducción de los protocolos de comunicaciones, explica como se fueron desarrollando las primeras redes y el proceso de estandarización de los protocolos.

El **Capítulo 2** nos da una definición clara y sencilla de lo que se entiende por protocolo y los aspectos fundamentales sobre como se estructuran.

En los **Capítulos 3 y 4** se exponen los pasos a seguir para detectar y llevar a cabo un buen control de errores y de flujo.

El **Capítulo 5** ofrece una visión parcial del protocolo, que se centrará en obtener un juego de reglas consistente y completo, es decir, un modelo de validación.

En el **Capítulo 6**, el lenguaje SDL nos permite desarrollar modelos de validación, y una vez diseñado un modelo, especificar de manera clara lo que entendemos por un diseño correcto.

En el **Capítulo 7** se expone una aplicación sobre lo visto en estructuración y especificación.

Por último, el **Capítulo 8** nos muestra que un protocolo en un nivel bajo de abstracción, puede verse como una máquina de estado finito, y éstas son muy importantes en tres áreas del diseño de protocolos: validación formal, síntesis de protocolos y test de conformidad.

OBJETIVOS

El objetivo fundamental que se pretende conseguir es el disponer de una guía básica sobre la Ingeniería de Protocolos para ser usada como primer contacto con la asignatura y como apoyo docente básico. Esta asignatura se enmarca dentro del área de Ingeniería Telemática.

MARCO DE PRESENTACIÓN DE LA INFORMACIÓN

Para que el acceso a este documento por parte del alumno sea rápido y sencillo, hemos hecho uso del lenguaje XML y antes de nada conviene repasar su historia y precedentes.

La versión 1.0 del lenguaje XML es una recomendación del W3C desde Febrero de 1998, pero se ha trabajado en ella desde hace unos años. Está basada en el anterior estándar SGML (Standard Generalized Markup Language, ISO 8879), que data de 1986, pero que empezó a gestarse desde principios de los años 70, y a su vez basado en el GML creado por IBM en 1969. Esto significa que aunque XML pueda parecer moderno, sus conceptos están más que asentados y aceptados de forma amplia. Está además asociado a la recomendación del W3C DOM (Document Object Model), aprobado también en 1998. Éste no es más que un modelo de objetos (en forma de API) que permite acceder a las diferentes partes que pueden componer un documento XML o HTML.

SGML proporciona un modo consistente y preciso de aplicar etiquetas para describir las partes que componen un documento, permitiendo además el intercambio de documentos entre diferentes plataformas. Sin embargo, el problema que se atribuye a SGML es su excesiva dificultad; basta con pensar que la recomendación ocupa unas 400 páginas.

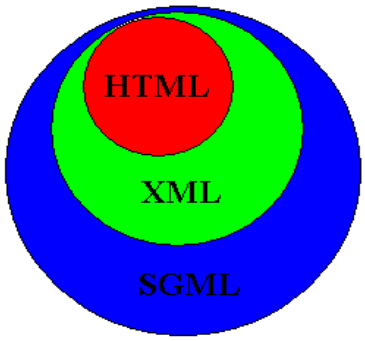
Así que, manteniendo su misma filosofía, de él se derivó XML como subconjunto simplificado, eliminando las partes más engorrosas y menos útiles. Como su padre, y este es un aspecto importante, XML es un metalenguaje, es decir:

1. Es un lenguaje para definir lenguajes.
2. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades.
3. Aprovecha las innegables ventajas del HTML pero a su vez permite realizar muchas más cosas.

Los elementos que lo componen pueden dar información sobre lo que contienen, no necesariamente sobre su estructura física o presentación, como ocurre en HTML.

Usando SGML, por otro lado, se definió precisamente el HTML, el lenguaje que nos es tan conocido. Entonces, ¿cuál es la diferencia entre ambos?

En una primera aproximación se puede decir que mediante XML también podríamos definir el HTML, con lo que podríamos considerar los siguientes conjuntos:



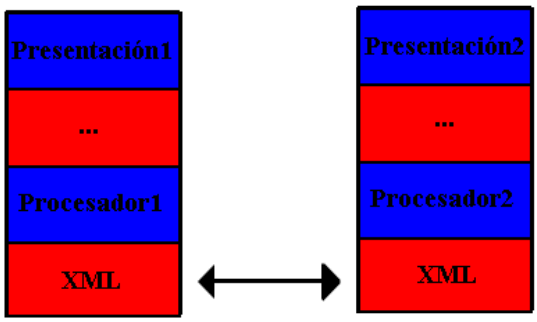
Durante el año 1998 XML ha tenido un crecimiento exponencial, y con ello me refiero sobre todo a sus apariciones en los medios de comunicación de todo tipo, menciones en páginas web, soporte software, tutoriales, etc.

De este modo entró en 1999 como si se hubiera llegado al final de una pista de despegue; y ahora XML por fin ha despegado. Y esto significa que ha pasado de ser una mera especulación a ser una realidad empresarial palpable y mesurable: los programas que lo soportan han crecido del mismo modo exponencial, y a día de hoy no hay empresa de software que se precie que no anuncie la compatibilidad de sus productos más vendidos con este nuevo estándar: Microsoft (Office 2000), Oracle (Oracle 8i, Web Application Server) o Lotus (Notes) son tres claros ejemplos de ello. Aún más increíble es pensar que hay empresas que se han creado entorno a él, u otras que han movido su actividad hacia su ámbito (de SGML a XML, por ejemplo, como ArborText).

Entonces, ¿será XML el sustituto de HTML?
No, básicamente XML no ha nacido sólo para su aplicación en Internet, sino que se propone como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, y casi cualquier cosa que podamos pensar.

No lo sustituirá, pero, aplicado en Internet, sí va a mejorar algo de lo que HTML empezaba a adolecer desde hace un tiempo: establece un estándar fijo al que atenerse, y separa el contenido de su presentación.

Se puede suponer de este modo que XML constituye la capa más baja dentro del nivel de aplicación, sobre el que se puede montar cualquier estructura de tratamiento de documentos, hasta llegar a la presentación. Y así podemos ver la compartición de documentos entre dos aplicaciones como intercambio de datos a ese nivel:



Pero ahora, y antes de seguir divagando sobre sus posibilidades, explicaremos los conceptos en los que se basa y la teoría de su aplicación, para poder entender nuestro documento.

Lo primero que debemos saber es que hay dos tipos de documentos XML.

- *Bien formados:* son todos los que cumplen las especificaciones del lenguaje respecto a unas reglas sintácticas, sin estar sujetos a unos elementos fijados en una DTD. De hecho 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 DTD: sus elementos y sobre todo la estructura jerárquica que define la DTD, además de los atributos, deben ajustarse a lo que la DTD dicte.

¿QUÉ ES LA DTD?

Las DTDs no son más que definiciones de los elementos que puede contener un documento XML, de la forma en que debe hacerlo y los atributos que se le pueden dar.

La DTD define la gramática de un documento XML. Es “opcional”, es decir un documento XML puede o no tener asociada una DTD, pero en caso que se realice la asociación, esta realiza la validación de las reglas que se han declarado.
En nuestro caso no existe DTD asociada, pero se dará una breve explicación para una mayor comprensión.

Cuando un documento cumple con las normas de escritura dadas por XML, se dice que el documento está bien formado. Adicionalmente cuando un documento está acorde con su DTD decimos que el documento es válido. Los browsers, tales como Netscape, MS Internet Explorer y otros, verificarán los dos aspectos.

Hay varios modos de referenciar una DTD en un documento XML:

1. Incluir dentro del documento una referencia al documento DTD en forma de URI (Universal Resource Identifier, o identificador universal de recursos).

   Por ejemplo:

   ```
   <!DOCTYPE ficha SYSTEM
   "http:// www.dat.etsit.upct.es / abarbero/DTD/ficha.dtd">
   ```

   En este caso la palabra SYSTEM indica que el DTD se obtendrá a partir de un elemento externo al documento e indicado por el URI que lo sigue, por supuesto entrecomillado.

2. O bien incluir dentro del propio documento el DTD.

   Por ejemplo:

   ```
   <?xml versión="1.0"?>
   <!DOCTYPE ficha [
   <!ELEMENT ficha (nombre+, apellido+, direccion+ foto?)>
    <!ELEMENT nombre (#PCDATA)>
    <!ATTLIST nombre sexo (masculino|femenino) #IMPLIED>
    <!ELEMENT apellido (#PCDATA)>
    <!ELEMENT direccion (#PCDATA)>
    <!ELEMENT foto EMPTY>
    ]
         <ficha>
         <nombre>Carolina</nombre>
         <apellido>Mercader</apellido>
         <direccion>c/ Ángel Bruna, 52</direccion>
         </ficha>
   ```

La forma de incluir el DTD directamente como en este ejemplo pasa por añadir a la declaración <!DOCTYPE y después del nombre del tipo de documento, en vez de la URI del DTD, el propio DTD entre los símbolos '[‘y’]'. Todo lo que hay entre ellos será considerado parte del DTD.

En cuanto a la definición de los elementos, después de la cláusula <!ELEMENT se incluye el nombre del elemento (el que luego se indicará en la etiqueta), y después en función del elemento:

- #PCDATA → Indica datos de tipo texto que son los más habituales.
- EMPTY → Indica que es un elemento vacío.

Para la definición de los atributos, se usa la declaración <!ATTLIST, seguida de:

- El nombre del elemento del que estamos declarando los atributos.
- El nombre del atributo.
- Los posibles valores del atributo.
- De forma opcional y entrecomillado, un valor por defecto del atributo si no se incluye otro en la declaración.
- Por último, especificar si es:

    - #REQUIRED → si es obligatorio.
    - #IMPLIED → si es opcional.
    - #FIXED → si el valor del atributo se tiene que mantener fijo a lo largo de todo el documento.

A la hora de indicar los elementos descendientes (los que están entre paréntesis) vamos que van seguidos de unos caracteres especiales, que sirven para indicar qué tipo de uso se permite hacer de estos elementos dentro del documento:

- **+** : uso obligatorio y múltiple; permite uno o más elementos de ese tipo dentro del elemento padre, pero como mínimo uno.
- ***** : opcional y múltiple; puede no haber ninguna ocurrencia, una o varias.
- **?** : opcional pero singular; puede no haber ninguno o como mucho uno.
- **|** : equivale a un OR, es decir, da la opción de usar un elemento de entre los que forman la expresión, y solo uno.

En todos los documentos XML, lo primero que tenemos que observar es la primera línea, ya que con ella deben empezar todos los documentos XML, porque es la que indica que lo que la sigue es XML y aunque es opcional, es más que recomendable incluirla siempre.

Puede tener varios atributos (los campos que van dentro de la declaración), algunos obligatorios y otros no:

- *versión:* indica la versión de XML usada en el documento. La actual es la versión 1.0, con lo que no debe haber mucho problema. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.

- *encoding:* indica la forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser el entender o no la codificación. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1 (ésta es la elegida por nosotros), etc. No es obligatorio salvo que sea un documento externo a otro principal.

- *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.

En cuanto a la sintaxis del documento hay que resaltar algunos detalles importantes:

- Los documentos XML son sensibles a mayúsculas, esto es, en ellos se diferencian las mayúsculas de las minúsculas. Por ello <FICHA> sería una etiqueta diferente a <ficha>.

- Además todos los espacios y retornos de carro se tienen en cuenta (dentro de las etiquetas, en los elementos).

- Hay algunos caracteres especiales reservados, que forman parte de la sintaxis de XML: <, >, &, " y '. En su lugar cuando queramos representarlos deberemos usar las entidades <, >, &, " y ' respectivamente.

- Los valores de los atributos de todas las etiquetas deben ir siempre entrecomillados. Son válidas las dobles comillas (") y la comilla simple (').

- Sólo puede haber un elemento raíz. El resto de las etiquetas deben colocarse dentro de este elemento principal.

Pasando al contenido en sí, las etiquetas nos pueden recordar a las de HTML. Es importante diferenciar entre elementos y etiquetas:

- Los *elementos* son las entidades en sí, lo que tiene contenido.
- Las *etiquetas* sólo describen a los elementos.

Un documento XML está compuesto por elementos, y en su sintaxis éstos se nombran mediante etiquetas.
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.

- Los elementos vacíos son aquellos que no tienen contenido dentro del documento.

Para saber qué nos depara el futuro y qué podremos obtener de XML como lenguaje de publicación en Internet, hace falta conocer de qué forma soportaran los navegadores el nuevo lenguaje.

Hay que tener en cuenta que XML es un lenguaje capaz de describir el contenido de la página, no su aspecto.

¿Y cómo un navegador puede enseñar algo de lo que no sabe su aspecto?
La idea es asociar al documento XML una hoja de estilo. Se puede hacer con CSS (Cascading Style Sheets) y con XSL (eXtensible StyleSheets Language), que permite añadir lógica de procesamiento a la hoja de estilo, en forma de lenguaje de programación (una especie de script). De este modo, dependiendo de la hoja de estilo que asociemos al documento o de la lógica que incluyamos, se podrá visualizar en cualquier plataforma: PalmPC, IE5, Netscape,… y con el aspecto (colores, fuentes,…) que queramos.

El navegador que hemos usado es el de Microsoft, el Explorer 5.0 que incorpora internamente todo lo anterior, permitiendo incluir documentos XML dentro de los clásicos HTML (mediante tags <XML> </XML>), o directamente leyendo documentos *.xml; y por otro lado permite el uso con ellas de hojas de estilo CSS y XSL.

El tipo de hoja de estilo que se ha asociado a nuestro documento XML es CSS. Los documentos CSS, o plantillas CSS, son descripciones detalladas del formato de las entidades XML. Son muy similares a las plantillas CSS utilizadas en HTML, ya que utilizan la misma normativa, pero tienen algunas diferencias, justificadas por el distinto enfoque de las etiquetas HTML (prefijadas de antemano y con algunos criterios definidos) y XML (libres e indefinidas por completo).

Las plantillas CSS pueden insertarse en el propio documento XML, pero lo más adecuado y recomendable es que formen parte de un archivo externo, con extensión .css, al que se hace referencia desde el documento principal. Este sistema recomendado tiene la gran ventaja de asegurar que, cada vez que se realice alguna modificación, todos los documentos XML relacionados, quedan inmediatamente actualizados. En el caso de estar incluidos en el mismo documento XML, cada modificación debe ser realizada en cada documento.

No exigen cabecera especial, ya que solo precisan las descripciones de los formatos de las etiquetas, que se denominan reglas, dispuestas con cualquier ordenación, no siendo problema el que existan errores de etiquetas o de atributos, e incluso que falten o sobren etiquetas, ya que todo código erróneo o sobrante es automáticamente ignorado.

Reglas de estilo

Los documentos CSS se componen de reglas de estilo. Una regla de estilo se compone de dos partes principales:

- El *selector*, que es el elemento que aparece en primer lugar (a la izquierda).
- La *declaración*, que es todo lo que figura a la derecha del selector, entre signos de "llaves" ({ y }).

La declaración simple se compone de una pareja de entidades. Estas entidades pueden ser la propiedad o el valor, situadas a la izquierda y derecha, respectivamente, del signo de "dos puntos" (:).

En muchas ocasiones se repiten los datos de la declaración, por lo que es muy práctico y útil agruparlos adecuadamente. Los selectores se pueden agrupar en listas separados por el signo de la "coma" (,).

La mayor parte de la información de los documentos web se encuentra en forma de textos, por lo que existen muchas propiedades CSS para su control. A continuación se muestra un breve resumen:

Propiedad	**Comentario**
Display:	Indica cómo visualizar el contenido. Cuando su valor es **block** se trata como un bloque y se termina con un "punto y aparte".
Font-family:	Indica el tipo de letra que se debe aplicar a su contenido. **Verdana** es una de las más utilizadas en la Web.
Font-size:	Indica el tamaño de letra que se debe aplicar a su contenido. Si la unidad es **pt**, el tamaño se fija en "puntos tipográficos".
Color:	Indica el color que se desea aplicar a su contenido. En este caso se han utilizado descripciones por nombre.
Text-align:	Indica la alineación del texto. Los valores **left**, **right** y **justify** significan "izquierda", "derecha" y "justificado".
Font-style:	Indica el estilo de letra que hay que aplicar al texto. Con el valor **italic** se selecciona el estilo en "cursiva".

Los colores pueden definirse en CSS por dos sistemas principales:

- Por su **nombre**
- Por su **descripción RGB**.

Cuando se utilizan colores sencillos es más fácil definirlos por su nombre, estando normalizados los siguientes dieciséis colores: **aqua** (azul claro), **black** (negro), **blue** (azul), **fuchsia** (fucsia), **gray** (gris), **green** (verde), **lime** (verde lima), **maroon**

(marrón) **navy** (azul oscuro), **olive** (verde oliva), **purple** (morado), **red** (rojo), **silver** (plata), **teal** (verde azulado), **white** (blanco) y **yellow** (amarillo).

La descripción de colores a través del modelo RGB (*Red-Green-Blue*, rojo-verde-azul), es decir, el patrón que se basa en la combinación de los tres colores primarios de luz, es más adecuada para cuando se desea precisar el color con total exactitud. Se puede utilizar la notación **decimal** (tres valores entre 0 y 255) o **hexadecimal** (tres valores entre 00 y FF), indistintamente, según interese.
Cuando se define un color por sus valores decimales RGB, hay que utilizar el modelo **color: rgb(***ValorRed***,***ValorGreen***,***ValorBlue***)**.

Para introducir comentarios entre las reglas del código CSS se utiliza el modelo

/ *comentarios*****/***

Los caracteres de comentarios no deben confundirse con los caracteres que se utilizan para ocultar las reglas de estilo a los navegadores que no soportan esta tecnología. Estos conocidos limitadores son **<!--** y **-->**.

Vamos a ver un breve ejemplo de lo que sería una hoja de estilo:

Nombre{font-family:Arial;color:black;font-size:12pt}

Apellido{font-family:Verdana;color:red;font-size:12pt}

Para insertar las imágenes se ha hecho uso del Microsoft PowerPoint, Microsoft Photo Editor y del Paint.

Bueno, este es un breve resumen de cómo se ha ido desarrollando el código de nuestro proyecto, si te ha gustado, existen infinidad de tutoriales, los cuales te pueden guiar para conseguir resultados extraordinarios.

Mi opinión sobre el futuro de XML es que:
- No todos lo utilizarán.
- Serán los grandes autores de páginas los más beneficiados.
- La curva de aprendizaje se pagará en menos horas de mantenimiento.
- Parte del futuro del intercambio de datos está en el XML, con lo que no ofrecer soporte para él es arriesgado.
- Las comunidades virtuales crearán DTDs de acuerdo a sus necesidades.
- Es probable que algún día los procesadores de texto almacenen texto en XML en lugar de formatos propios.
- XML probablemente reemplazará a SGML.

Las conclusiones finales de este proceso de presentación de información se reducen a dos realmente importantes:

- Es fácil crear una página web a partir de un documento XML y de una plantilla CSS.
- Si CSS no proporciona fácilmente alguna solución, se puede recurrir al HTML.

No obstante, hay que dejar claro que este proceso de presentación de información con XML-CSS, solo está indicado para exponer documentos que incluyan principalmente textos y datos expresados en forma secuencial, o sea, información que deba ser mostrada en su totalidad, abarcando cualquier libro, informe, revista, memoria, artículo, folleto, currículum, exposición, crítica, cuento, noticia, examen, normativa, catálogo, etc., incluyendo textos, datos numéricos, hiperenlaces, tablas, listas e imágenes, principalmente.