

Xhtml: reformular html en xml

Por Antonio de la Rosa

EN AGOSTO PASADO *xhtml* (*extensible hypertext markup language*) alcanzó el estatus de recomendación aprobada por el W3C, aunque la idea de reformular html en xml ya existía desde hacía algún tiempo en este organismo con *Voyager*. El nombre es bastante revelador respecto a sus intenciones: “viajar” desde uno hasta otro sin perder en la transición todo el esfuerzo realizado hasta ahora en el desarrollo de html.

En alguno de los antiguos borradores de trabajo se recogen sus principales directrices: **modularización** de html mediante la especificación de conjuntos bien definidos (bien formados) de etiquetas y uso de **perfiles de documentos**, una serie de *DTD's* xml —que se corresponden con entidades abstractas llamadas *namespaces*— contra las que se pueda validar documentos html.

«Xhtml es un glosario para xml compuesto por elementos y atributos incluidos en html 4.0, pero que usan una sintaxis xml en lugar de una sgml»

Xhtml 1.0 es el intento del W3C de reescribir html 4.0 en xml 1.0. Esto implica fijar estrictamente la laxitud de su sintaxis, y que se requieran etiquetas finales para elementos que en html se omiten:

— `<p>` pasa a `<p></p>`.

— Los elementos vacíos como `<hr>` o `
` deben finalizar con `</>` en lugar de simplemente `>`.

— Los valores de los atributos han de aparecer entre comillas: `` en vez de ``.

— Los nombres de elementos y atributos deben escribirse en minúscula y siempre han de estar correctamente anidados (`<p></p>` y no `<p></p>`).

Un ejemplo de aplicación práctica de sintaxis *xhtml* se puede encontrar en el código de la especificación localizada en una de las páginas de W3C.

<http://www.w3.org/TR/xhtml1/>

Xhtml pretende hacer de los documentos html que sean algo más que tipos xml bien creados. En este lenguaje, en general, hay dos categorías: bien formados, si obedecen su sintaxis, —lo que significa que deben cumplir una serie de reglas de producción *Ebnf* (*extended backus-naur form*)—, y válidos, cuando atienden a las restricciones de la *DTD* a la que pertenecen. Lo que *xhtml* intenta es proporcionar a html una *DTD* contra la que se puedan validar los documentos. De hecho, propone tres: una estricta para los nuevos, otra de transición para viejos documentos html convertidos, que todavía usen etiquetas desechadas como `<applet>`, y una tercera para aquellos que usen frames.

Simplificando la cuestión, *xhtml* es un glosario para xml compuesto por elementos y atributos incluidos en html 4.0, pero que usan una sintaxis xml en lugar de una *sgml* (recordemos que html, de hecho, no es más que una *DTD sgml*). Y se supone que la van a aplicar estrictamente.

En general, la mayoría de documentos html que circulan actualmente por el www no son válidos por la sencilla razón de que no necesitan serlo para ser funcionales. Hasta hace poco, gran parte de la rivalidad entre *Netscape* y *Micro-*

soft, como principales fabricantes de navegadores, giraba en torno a la cantidad de código html inválido que podían procesar, reconstruir y presentar aceptablemente. Por ejemplo, *Ie* (el ganador) rellena los documentos con `</table>` si falta esta etiqueta, mientras que *Netscape navigator* no lo hace. Consecuentemente, muchas páginas del web de *Microsoft* a las que les falta no pueden verse con el segundo navegador (se deja a juicio del lector si se trata o no de un sabotaje deliberado). En cualquier caso, si en esta página se requiriera html válido, esto no ocurriría.

«Hasta hace poco, gran parte de la rivalidad entre Netscape y Microsoft giraba en torno a la cantidad de código html inválido que podían procesar, reconstruir y presentar aceptablemente»

Desde el punto de vista de los diseñadores web y de las empresas que los contratan, es extremadamente difícil optimizar este lenguaje teniendo en cuenta todos los navegadores existentes (ya es complicado hacerlo sólo con los dos más populares). Esta situación supone o bien una gran pérdida de tiempo y recursos, o limitarse siempre a diseñar para uno en concreto renunciando a las posibilidades de los otros. En un mundo ideal cualquier usuario debería simplemente saber escribir html para asegurar que sus páginas van a aparecer correctamente con cualquiera de ellos.

Un motivo para el desarrollo de *xhtml* es proporcionar interoperabilidad, lo que significa que se pueda acceder al código con resul-

tados aceptables desde cualquier navegador, agente o interfaz. Esta idea tiene presentes las previsiones del W3C: en el 2002 el 75% del acceso al www se producirá desde lo que hoy consideramos plataformas alternativas: televisión, teléfonos móviles, etc.

Además, dada la creciente emergencia de todo tipo de lenguajes de etiquetas (esa plaga de acrónimos que acaban en “ml”), es lógico pensar que en la comunidad www existe siempre el deseo de ampliar los límites de la norma. En xml es relativamente fácil introducir nuevos elementos o atributos. *Xhtml* pretende servir de marco a la combinación entre elementos existentes y nuevos (siempre que respete una sintaxis, la de xml), lo que supone una ventaja tanto para el desarrollo de contenido como para el diseño de nuevos navegadores. En otras palabras, se intenta superar la tradicional ineptitud de html para gestionar contenido.

«El concepto central de *xhtml* es la noción de namespaces: entidades abstractas que sirven para identificar elementos»

Por otra parte, *xhtml* se puede considerar una de las iniciativas — junto a *DOM* (*document object model*), el resto de “mls”, la proyectada nueva versión del protocolo http (*http-ng*) y una serie de tecnologías emergentes: *WebDAV* (*www distributed authoring and versioning*), *Jini*, bases de datos relacionales-orientadas a objetos y *SQL3*, *Jdbc 2.0* (*java database connectivity*), *Sqlj* (*embedded SQL for java*), *Dasl* (*DAV searching and locating*), *JavaMail*, etc.— que tratan de reconducir la disposición del www hacia un entorno universal orientado a objetos en el cual, sobre ciertos principios de desarrollo comunes, se pueda utilizar una o varias de las

tecnologías existentes para responder a problemas concretos, sin que ello implique tener que renunciar a las demás.

Lo que realmente significa la aparición de *xhtml*, a corto plazo, es que el grupo de trabajo html del W3C ha decidido abandonar sus planes originales de reducir progresivamente su presencia en el www en beneficio de xml y reformular html 4.0 de una forma compatible con éste. En otras palabras, tender un puente, amortizar el esfuerzo realizado en el desarrollo de html y potenciar las tres DTDs antes mencionadas. Cada una de ellas modulariza html 4.0 en subconjuntos de elementos: *xhtml1-meta.mod* (metadatos), *xhtml1-tables.mod* (tablas), *xhtml1-form.mod* (formularios), *xhtml1-linking.mod* (enlaces), *xhtml1-events.mod* (eventos), *xhtml1-script.mod* (*scripting*) o *xhtml1-style.mod* (hojas de estilo). En total existen 28 módulos y tres conjuntos de entidades (símbolos, latín-1 y caracteres especiales) aunque las tres DTDs *xhtml* no los implementan todos (la estricta, por ejemplo, sólo utiliza 22).

«Un motivo para el desarrollo de *xhtml* es proporcionar interoperabilidad y que se pueda acceder al código con resultados aceptables desde cualquier navegador»

Las DTDs principales se enlazan con los módulos por medio de entidades —recordemos que en xml se usan éstas y sus declaraciones para asociar un nombre con algún otro fragmento del documento: datos regulares, partes de DTDs o archivos externos—. Tomemos como ejemplo la parte de la DTD estricta que hace referencia al módulo de los elementos de enlace:

```
<!-- linking module->
<!entity % xhtml1-linking.module
“include”>
<![%xhtml1-linking.module:[
<!entity % xhtml1-linking.mod
public “-//W3C//elements xhtml
1.0 linking//en” “xhtml1-
linking.mod”>
%xhtml1-linking.mod;
]]>
El módulo de los elementos de
enlace al que se hace referencia es:
<!--...->
<!-- xhtml 1.0 linking module>
<!-- file: xhtml1-linking.mod>
<!-- d2. linking a, base, link->
<!-- anchor element>
<!entity % shape
“(rect/circle/poly/default)”>
<!entity % coords “cdata”>
<!entity % a.content “( #pcdata /
%inline-noa.mix;)*”>
<!element % a %a.content;>
<!attlist a
%common.attrib;
name cdata #implied
href %URI; #implied
%alink.attrib;
charset %charset;
#implied
type %contenttype;
#implied
hreflang %languagecode;
#implied
rel %linktypes;
#implied
rev %linktypes; #implied
accesskey %character;
#implied
tabindex %number;
#implied
>
```

```

<!-- base element>
<!entity % base.content
“empty”>
<!element base %base.content;>
<!attlist base
  href %uri; #required
>
<!-- link element>
<!entity % link.content “empty”>
<!element link %link.content;>
<!attlist link
%common.attrib;
  href %uri;
#implied
  charset %charset;
#implied
  type %contenttype;
#implied
  hreflang %languagecode;
#implied
  rel %linktypes;
#implied
  rev %linktypes;
#implied
  media %mediadesc;
#implied
>
<!-- end of xhtml1-linking.mod->

```

Si se elige como ejemplo la lista de atributos del elemento `<a>` y nos remitimos a la especificación html 4.0, se observa a primera vista que *xhtml* reduce su número eliminando algunos: *class*, *datafld*, *datasrc*, *dir*, *id*, *methods*, *style*, *target*, *title* y *urn*. En consecuencia, ¿se reduce con *xhtml* el alcance de html sin alcanzar las posibilidades de xml? En el caso de los enlaces ¿las posibilidades siguen siendo las mismas, ningún intento de implementar soluciones *Xlink* o *Xpointer*? La respuesta a estas preguntas es que, como siempre, la especificación propone y el software dispone, la funcionalidad de los

documentos html, xml o *xhtml* es un asunto que depende exclusivamente del programa que los interprete.

Además de la modularización de html, el concepto central de *xhtml* es la noción de *namespaces*: entidades abstractas que sirven para identificar elementos (html, xml, *xhtml*, sgml, etc.) dentro de un contexto concreto. Es uno de esos conceptos que es más fácil explicar en la práctica: por ejemplo, las tres DTDs *xhtml* mencionadas anteriormente (estricta, de transición y la que se aplica a documentos con frames) son tres *namespaces*. Dentro de cada una de ellas se puede encontrar, por ejemplo, el elemento `` con sus atributos. En teoría podría significar cosas diferentes o comportarse de forma distinta —si el elemento es html: significa (el browser lo interpreta), si es xml sólo significa sintácticamente, no semánticamente (un elemento xml podría ser la etiqueta `<blabla></blabla>`)—. Este problema puede solucionarse haciendo que cada instancia del elemento en un documento se refiera a su propio *namespace*.

Pongamos un ejemplo práctico en *xhtml*:

```

<html
xmlns="http://www.w3.org/TR/xhtml1/strict" xml:lang="en"
lang="en">
<head>
  <title>A Math
example</title>
</head>
<body>
  <p>Lo siguiente son etiquetas
MathML:</p>
  <math
xmlns="http://www.w3.org/TR/REC-MathML">
    <apply> <log/>
      <logbase>

```

```

<cn> 3
</cn>
</logbase>
<ci> x </ci>
</apply>
</math>
</body>
</html>

```

En este caso se puede comprobar cómo desde el mismo documento se pueden referenciar *namespaces* distintos, lo que implica que se utilizan elementos de diferentes contextos (en el ejemplo, elementos *xhtml* y *MathML*) enlazados mediante el atributo *xmlns* (*xml name space*). La función de los *namespaces* es permitir que se escriban programas capaces de encontrar siempre los elementos de información para los que han sido diseñados (por ejemplo el elemento `<blabla></blabla>`) incluso en un entorno distribuido y heterogéneo.

<http://www.w3.org/TR/REC-xml-names>

«En el 2002 el 75% del acceso al www se producirá desde lo que hoy consideramos plataformas alternativas: televisión, teléfonos móviles, etc.»

En definitiva, lo que intenta *xhtml* es hallar un término medio entre ambos lenguajes, lo que parece una solución razonable para el período de convivencia que les queda a estas dos normas. La cuestión clave es: ¿cuánto cuesta hacer que documentos html sean compatibles con la norma xml? Hasta el momento ambos estándares seguían sus respectivos caminos sin demasiadas interferencias: a los parsers xml no les importa procesar etiquetas html mientras estén bien formadas, y el software diseñado

para éste simplemente ignora todo lo que no entiende. En cuanto al problema de formato vs. contenido, se está solucionando mediante el empleo cada vez más frecuente de hojas de estilo (especificación *css2*, implementación de *xsl* en *l_e 5*, etc.).

La aparición y utilidad de *xhtml*, desde mi punto de vista, se debe a la tendencia generalizada del *www* a modularizarse, estructurarse y profundizar en una serie de buenas ideas —*namespaces*, localizadores de recursos, transclusión-inclusión, enlaces más sofisticados, separación de contenido, formato y comportamiento, tendencia hacia la orientación a objetos, acercamiento de los lenguajes de formato y los de programación, definición de tipos de datos, herencia entre elementos o simplemente documentos contruidos con más inteligencia, etc.— que están en el aire desde hace tiempo y ahora empiezan a ser utilizadas con fines prácticos. En mi opinión, *xhtml* aspira a ser un medio más para que *html* sea capaz de hacerlas realidad; la pregunta es si se trata de una solución para alargar la vida de *html* o una forma de intentar que evolucione hacia donde el *www* está llegando.

Bibliografía

Harold, Elliotte Rusty. *Xml bible*. New York: IDG books worldwide inc., 1999. Isbn: 0-7645-3236-7.

Reformulating html in xml. W3C working draft 5th. December 1998.

<http://www.w3.org/TR/WD-html-in-xml>

Xhtml 1.0: the extensible hypertext markup language: a reformulation of html 4.0 in xml 1.0.

W3C proposed recommendation.

<http://www.w3.org/TR/xhtml1/>

Antonio de la Rosa, consultor/investigador, *Wisdom. Nieuwe Herengracht, 113 1011SB, Amsterdam. Holanda.*

antonio@wisdom.nl