Lenguaje XML

Fecha: 15-7-08 Hora: 22: 53 : 47

XML está diseñado "para hacer fácil y directo el uso de SGML en la Web: fácil de definir tipos de documentos, facilidad para el autor y gestor de documentos definidos en SGML, y fácil de transmitir y compartirlos a través de la Web." Define "un dialecto extremadamente simple de SGML, el cual está completamente descrito en la Especificación XML. El objetivo es habilitar que el SGML genérico sea servido, recibido, y procesado en la Web de la forma que ahora permite hacerlo HTML." "Por esta razón, XML ha sido diseñado para facilitar la implementación, y para permitir la interoperatividad entre SGML y HTML" [citas de la especificación XML].

Los documentos XML pueden empezar con un prólogo, en el que esencialmente se define:

- Una declaración XML
- Una declaración de tipo de documento

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE documento [
<!ELEMENT documento (p | imagen | ejemplo)*>
<!ELEMENT p (#PCDATA|destacar)*>
<!ELEMENT destacar (#PCDATA)>
<!ATTLIST destacar
               importancia CDATA #REQUIRED>
<!ELEMENT imagen EMPTY>
<!ATTLIST imagen
               fichero CDATA #REQUIRED>
<!ELEMENT ejemplo (#PCDATA)>
]>
```

En la declaración XML, <?xml version="1.0" encoding="UTF-8"?> Indicamos:

- Información sobre la versión de XML que estamos utilizando. Por el momento sólo puede ser la versión 1.0.
- Información sobre el tipo de codificación de carácteres que estamos utilizando. En nuestro caso es el código ASCII de 7 bits, que es un subconjunto del código Unicode denominado UTF-8. No hubiese sido necesario declararlo, ya que es el que los parsers manejan por defecto.

En la declaración del tipo de documento,

```
<!DOCTYPE documento [
<!ELEMENT documento (p | imagen | ejemplo)*>
<!ELEMENT p (#PCDATA|destacar)*>
<!ELEMENT destacar (#PCDATA)>
<!ATTLIST destacar
               importancia CDATA #REQUIRED>
<!ELEMENT imagen EMPTY>
<!ATTLIST imagen
               fichero CDATA #REQUIRED>
<!ELEMENT ejemplo (#PCDATA)>
```

```
]>
```

Asociamos la DTD respecto de la cual construimos el documento. En nuestro ejemplo va implícita en el propio documento XML, aunque también puede hacerse externa al documento e incluso de una forma mixta. Si la hubiésemos escrito en un fichero "ejemplo.dtd" tendríamos que referenciarla de la siguiente manera:

<!DOCTYPE documento SYSTEM "ejemplo.dtd">

Ambas partes del prólogo son opcionales, aunque en el caso de incluir ambas la declaración XML tiene que ir antes.

Otras construcciones de marcado.

Hasta aquí hemos visto los componentes más importantes de un documento XML. Aunque si observamos el ejemplo veremos que existen otras componentes que podemos utilizar:

Comentarios

Mediante los cuales podemos proporcionar información que el parser no tendrá en cuenta.

<!-- Esto es un comentario -->

Los comentarios empiezan con los caracteres "<!--" y terminan con "-->" y pueden colocarse en cualquier sitio excepto dentro de las declaraciones, etiquetas y otros comentarios.

CDATA

Permiten integrar texto en un documento en XML que de otra forma sería interpretado como etiquetas. Es decir, estamos introduciendo texto que luego el procesador XML va a mostrar pero no va a procesar como marcado.

```
<![CDATA[
        Aqui puedo poner lo que quiera.
] ]>
```

Los CDATA empiezan con los caracteres "<![CDATA[" y termina con "]]>".

Dentro de ellos podemos colocar cualquier cosa ya que no va a ser interpretado, con la salvedad de la cadena que indica el final de CDATA, "]]>", ya que el procesador al encontrársela entendería que la sección CDATA ya ha terminado con las nefastas consecuencias que ésto puede tener.

Documentos bien formados y documentos válidos

Como ya he escrito anteriormente, a diferencia del SGML, no es necesario que un documento XML esté asociado a una DTD.

El documento XML con el que empezábamos el documento, lo podíamos haber escrito de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Esto es un comentario -->
<documento>
<p>Mi Primer <destacar importancia="1">documento XML
</destacar></p>
<p>Comienza con la etiqueta <documento></p>
<p>A continuacion colocamos un elemento sin contenido
</p>
<imagen fichero="imagen.gif"/>
<p>Y ahora una etiqueta CDATA.</p>
<ejemplo>
<![CDATA[
        Aqui puedo poner lo que quiera.
] ]>
</ejemplo>
</documento>
```

Y si lo pasásemos por un parser de XML no nos daría ningún error. Por tanto, en función de si lleva asociada una DTD o no, podemos diferenciar dos tipos de documentos XML:

- Válidos, aquellos que siguen las reglas de una DTD especifica.
- Bien formados (well-formed), que no tienen necesariamente una DTD asociada, pero siguen las reglas del XML al pie de la letra.

Evidentemente, los documentos válidos son bien formados.

Lenguaje XSLT

XSL son las hojas de estilo extensible del XML, en realidad están formadas por dos lenguajes: XSLT, lenguaje de transformación, y XSL-FO, de formateo, éste último no tiene un soporte que justifique su uso, al referirnos aquí a las XSL, estamos hablando de XSLT.

Aqui puede observar el aspecto visual de una XSLT, estas actúan transformando los archivos XML y pueden realizar búsquedas, ordenamientos, calculos matemáticos, lógicos, etc. como un lenguaje tipo PHP. ësta XSLT que observa debajo está controlando una base de datos de un catálogo de libros. Dentro de las XSLT también se escriben estilos CSS. Ésto no pretende enseñar XSLT, sino sólo, que usted tome contacto con ésta realidad e intuya sus posibilidades, que son enormes; considere tambien que XML y XSLT se pueden ejecutar en el servidor, allanando todo la problemática de las compatibilidades.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/TR/WD-xsl"
xmlns="http://www.w3.org/TR/REC-html40"
```

```
result-ns="">

<xsl:template match="/">
<HTML>
<HEAD>
<TITLE>Catálogo de libros</TITLE>
</HEAD>
<BODY>
<link rel="stylesheet" type="text/css"
href="catalogo.css" />
<H1 style="color:red">CATÁLOGO DE LIBROS</H1>
<P>Editoriales ordenadas por orden alfabético:</P>
<xsl:for-each select="//libro" order-by="+Editorial">
<B><xsl:value-of select="Editorial" /></B><BR/>
</xsl:for-each>
<P>Títulos de libros ordenados por orden alfabético
de sus editoriales:</P>
<xsl:for-each select="//libro"
order-by="+Editorial;+Título">
<B><xsl:value-of select="Título" /></B>
(<xsl:value-of select="Editorial" />)<BR/>
</xsl:for-each>
<P>Títulos de libros ordenados por sus ISBN
(de mayor a menor):</P>
<xsl:for-each select="//libro" order-by="-ISBN">
<B><xsl:value-of select="Título" /></B>
(<xsl:value-of select="ISBN" />)<BR/>
</xsl:for-each>
<P>Títulos de libros ordenados por sus páginas
(de menor a mayor):</P>
<xsl:for-each select="//libro" order-by="+Páginas">
<B><xsl:value-of select="Título" /></B>
(<xsl:value-of select="Páginas" />)<BR/>
</xsl:for-each>
<xsl:comment>Esto es un comentario
sobre programación.
</xsl:comment>
<xsl:element name="Programador">Programador:
Carlos González</xsl:element>
</BODY>
</HTML>
</xsl:template>

</xsl:stylesheet>
```