PHP Simple HTML DOM Parser

Para abrir el mes de junio seguimos el hilo en nuestros artículos que buscan difundir el conocimiento libre del Patrimonio Tecnológico de la Humanidad. En anterior oportunidad estudiamos **PHP curl**, una herramienta basada en **cURL** y por ende en **libcurl** (¡ea, esto último *no lo mencionamos allá!).* Como en ese caso obtuvimos un método para descargar páginas web enteras, incluso si hay que pasarle datos con el método POST y/o hay que introducir usuario y contraseña. Esta entrada busca *extraer, analizar e incluso modificar dichos datos* ¡vente con nosotros!

https://twitter.com/Obijuan_cube/status/775301739432083456

PHP Simple HTML DOM Parser.

Presentación.

Buscábamos una "parser" o analizador código HTML y aunque **PHP** tiene su modelo establecido nos decantamos por algo más fácil pero con el inconveniente que debemos apoyar la licencia de uso del Instituto Tecnológico de Massachusetts ("MIT license") y que NO viene integrado al lenguaje PHP. Lo bueno es que apenas son 65037 bytes en un archivo con un guion escrito en lenguaje **PHP** para que funcione (la licencia nos obliga a distribuir completo los archivos de ejemplos y manuales, todos escritos en idioma inglés).

El analizador de HTML (y XML) que veremos y aprenderemos a usar tiene el pomposo nombre de "**PHP Simple HTML DOM Parser**" *con todas las implicaciones que derivan de llamar "simple" al* D.O.M. ("Document Object Model").

Document Object Model.

El Modelo de Objeto Documento es una norma propuesta por el Consorcio World Wide Web o **W3C** que son quienes hacen las recomendaciones y sientan las bases, de facto, del Web creado por el Doctor Tim Berners-Lee. En dicho papel de trabajo, a nuestro modo de ver las cosas, se propone que una página web es un documento y todo lo que está contenido en ella está representada por nodos. Si nos ponemos a ver tiene sentido porque la norma HTML5 establece unas sub divisiones que a su vez contienen títulos, sub títulos, párrafos, listas, etc.

Este concepto es mucho más amplio porque contempla, aparte de propiedades, también métodos que por definición HTML5 no contempla y para respetar la neutralidad hacen de la vista gorda del JavaScript (¿definirán algún día al CSS para suplantarlo? Sí, los sabemos, *nuestro pensamiento siempre es profano*).

El DOM está estructurado sobre tres pilares básicos:

- "Core DOM": modelo normativo para todos los tipos de documentos.
- "XML DOM": para documentos XML (febrero, 1998).
- "HTML DOM": no menos importante, aunque aparezca en tercer lugar, para documentos HTML.

Autores.

La idea original proviene de José Solorzano quien comenzó un proyecto bautizado como "HTML Parser for PHP 4" (dale con esto de los nombres largos ¿será que estoy leyendo muchos artículos en inglés y me estoy contagiando? Porque el idioma castellano es bastante prolijo...) que como podemos ver funciona con PHP 4 y pues va a ser que ya está descontinuado (incluso tienen un avisito recomendando el proyecto que vemos hoy acá).

Basado en ello S.C. Chen (me578022@gmail.com) programó una versión actualizada y basada en DOM con ayuda de Yousuke Kumakura y publicada en SourceForge, un alojador de contenido de software de código abierto que alberga más de 430.000 proyectos (3,7 millones de usuarios y 42 millones de clientes). Solo sabemos que S.C Chen es famoso a nivel mundial por su ópera prima que trajimos en este momento a colación.

Core DOM.

(pronto desarrollaremos este tema).

XML DOM.

Para poder entender el **HTML DOM** debemos primero estudiar y aprender el **XML DOM** y a continuación haremos nuestro racionamiento y sustento.

Concepto de XML.

"eXtensible Markup Language" o Lenguaje de Marcado Extensible (lo siento por el castellano pero lo seguiremos identificando como **XML** dado su impacto *de facto* en el mundo entero) es un lenguaje que "describe" como hacer las cosas y que además se puede ampliar en cualquier momento conservando una *retrocompatibilidad*. Fue diseñado para almacenar y transportar datos y que, además, pudiera ser leído por ordenadores y humanos por igual.

Si queréis leer sobre nuestra disertación acerca de lo que es un lenguaje de marcado, os invitamos a leer nuestro tutorial en línea sobre HTML. XML es guardado en ficheros codificados, generalmente, en UTF-8 para lograr compatibilidad con los diversos idiomas del mundo, nuestro tutorial sobre HTML5 también habla y describe al respecto en forma detallada.

Normas XML.

Hay unas normas destacadas que mencionaremos y que a futuro, en la medida que tengamos tiempo para ello, le publicaremos sus diferentes entradas en nuestro humilde blog.

- XML AJAX
- XML DOM
- XML XPath
- XML XSLT
- XML XQuery
- XML DTD
- XML Schema
- XML Services

¿Qué es y qué NO es XML?

Aparte de lo que ya definimos, hay otros detalles que decir acerca del XML: fue diseñado para ser autodescriptivo, ya que a diferencia del HTML tiene muy pocos "comandos" o etiquetas ***y el XML como tal no hace absolutamente nada por sí solo.*** Esa es entonces la principal diferencia con HTML, que se enfoca en *presentar* los datos mientras que XML se enfoca en *transportar* datos.

Decimos que se puede *extender* ya que nosotros mismo "inventamos" nuestros propias etiquetas, así que no tenemos limitación ***pero una vez hallamos agregado más etiquetas a una información preexistente ésta seguirá siendo completamente leída y manipulada por distintas aplicaciones sin ningún tipo de problema.***

Por supuesto, hacemos la advertencia que hablamos de **extender**, no de **contraer:** al eliminar algún dato *por supuesto que las aplicaciones ya no podrán realizar su trabajo completo, y en muchos casos se negarán a realizarlo de plano.*

Elementos de un documento XML.

Un documento XML está constituido por un elemento **raíz** el cual tiene elemento **ramas o nodos** que a su vez pueden contener subelementos y así sucesivamente. Cada elemento a su vez puede contener **atributos.**

De manera obligatoria un documento XML debe tener un elemento raíz, y aunque un prólogo **no** es obligatorio ni forma parte del documento es muy recomendable agregarlo; siempre va al principio del documento.

La función del prólogo es denotar la versión XML del documento y, de manera adicional, la codificación de caracteres el cual ya dijimos **UTF-8** es lo más recomendable. La sintaxis del

prólogo comienza con un símbolo "menor que" *o mejor dicho, un corchete angular de apertura, ya que no estamos hablando de matemáticas en este caso (pero ayuda muchísimo a entender el concepto de aperura y cierre).* Debe cerrar con un símbolo "mayor que" o corchete angular de cierre. Además de los dos símbolos anteriores se necesitan dos signos de interrogación de cierre y los contenidos deben estar entrecomillados ***para lo cual recomendamos las comillas simples, esto nos ahorra problemas para cuando vayamos a programar.*** Veamos un ejemplo de prólogo:

El prólogo es un caso especial, y ya dijimos que no forma parte del documento, por lo tanto tiene distintas reglas que las aplicadas a los elementos, los cuales pasamos a describir en la siguiente sección.

Sintaxis de los elementos XML.

- Cada uno de los elementos debe tener su correspondiente cierre pero con una barra invertida, por ejemplo:
    - texto
- Los nombres de los elementos distinguen mayúsculas de minúsculas por lo tanto no es igual a .
- Los nombres de los elementos **deben comenzar con una letra o en su defecto con un guion bajo " _ ".**
- Los nombres de los elementos **no pueden comenzar con "XML" o sus combinaciones de mayúsculas y/o minúsculas.**
- Los nombres de los elementos **no puden contener espacios** (de hecho un espacio denota que comienza un atributo, por eso no pueden contener espacios).
- Los subelementos deben estar correctamente anidados dentro de los elementos, ejemplo:
    - valor
- Los atributos de los elementos deben estar entrecomillados (lo mismo que dijimo para el prólogo se aplica en este caso), ejemplo:
    - dato
- Como tal vez hayan captado, hay unos caracteres especiales en los documentos XML que no podremos usar directamente porque se presta a confusión para los ordenadores a la hora de leer el documento XML **por lo tanto debemos sustituirlos por otros caracteres si queremos usarlos como datos**:
    - Corchete angular de apertura "" lo sustituimos por "**>**"(recordad "gt"= "greater than", mayor que).
    - *Ampersand* o "et" (proviene del latín, como en "etcétera"): "**&**" lo sustituimos por "**&**".
    - Apostrofo " **'** " lo susituimos por "**'**".

    - Comillas dobles " **"** " las sustituimos por "**"**" ("quotation mark").
- Si queremos o necesitamos dejar un comentario para nosotros los seres humanos y que sea ignorado por el ordenador hacemos lo mismo que en el lenguaje HTML, lo encerramos entre los siguientes signos:

    - *Nota:* dentro de un comentario **no podemos** escribir dos guiones juntos ya que dos guiones juntos indican inicio y fin de comentario, así que se prestaría a confusión para los ordenadores.
- En los documentos XML debemos tener en cuenta que todo espacio es considerado como tal, es decir, los espacios se conservan y son un dato en sí mismo y así será leído *ya quedará de parte de la aplicación el cortar o comprimir espacios "innecesarios". **Este comentario lo hacemos porque los documentos XML pueden ser leídos -y editados- por nosotros los humanos y un espacio en blanco mal colocado de nuestra parte hará que la aplicación que va a leer los datos le suceda una excepción pero nosotros no veamos el error al abrir el archivo nosotros mismos.***
- Parecerá una tontería pero el caracter que indica el final de una línea es el caracter **LF** o "Line Feed" OSEA al caracter ASCII 10 *y debemos tener siempre presente que en el sistema operativo Windows se utiliza **LF** más **CR** (ASCII 10 y 13 respectivamente) y en los viejos sistemas operativos Mac solo **CR**.*

Características de los elementos de un documento XML.

- Un elemento comienza desde el primer corchete angular de apertura hasta el último corchete angualr de cierre.
- Un elemento puede contener:
    - Nada (pues eso, vacío, ni siquiera un espacio -si tuviera un espacio no sería vacio-).
    - Ya que introducimos el concepto de elemento vacio -y esto debería estar en la sintaxis- un elemento *tácitamente vacio lo podremos denotar de la siguiente manera y ambos son iguales y válidos:*

    - Texto -datos, para los ordenadores-.
    - *Otros elementos -recordad anidar correctamente-.*
    - Cualquier combinación de los tres anteriores, las combinaciones son infinitas *y por ende decimos que son extensibles -y retrocompatibles-.*

Una breve aclaratoria sobre los atributos.

Hay que tener especial cuidado con los atributos que le coloquemos a los elementos. Para no caer en abstracciones les haremos un caso práctico: consideremos un elemento llamado "**persona**" y lo que eso significa para nosotros los castellanohablantes.

Ahora consideremos colocarle un atributo de género *femenino* o *masculino:*

```
        Pedro          María
```

Como vemos podemos "cargar" en memoria de ordenador solamente los hombres -o las mujeres- *pero la aplicación que lee dichoa rchivo debe estar programada, de manera previa, a buscar esos dos géneros solamente* ***¿Qué sucedería si surgiera un tercer tipo de sexo, como por ejemplo hermafrodita?***

```
        Pedro       Michelle          María
```

No habría forma ni manera que la aplicación pudiera "ampliarse" para que leyera datos adicionales ampliados, así que es una mejor idea plantearlo de la siguiente manera:

```
        masculino       Pedro           hermafrodita
    Michelle          femenino      María
```

Como veís, colocado de esta manera la aplicación puede ir generando una matriz con los distintos *nuevos tipos de valores tal como lo hace con los valores en sí mismos.* Aparte de esta ventaja debemos recordar también que los atributos NO pueden contener múltiples valores y tampoco pueden tener estructuras tipo árbol, *entonces* ***¿para que diantres sirven los atributos?*** Antes de responder esta pregunta queremos dejar en claro algo más sobre los atributos: los atributos son elemento *fuertemente tipados -como decimos en los lenguajes de programación-, son estructura rígidas que son difíciles de cambiar a futuro pero que debido a esto podremos tomar ventaja; veamos la siguiente sección.*

Eliminando ambigüedades en el nombrado de los elementos.

Como ya bien sabemos los documentos XML nos sirven para almacenar datos y debido a que nosotros los seres humanos asignamos un nombre para todo en nuestro universo -y a medida que lo seguimos descubriendo- no siempre escogemos nombres únicos. *Tomemos por caso la palabra "banco":* tal vez primero pensemos en una institución financiera **pero es que también un "banco"**

es un objeto sobre el cual podemos sentarnos.

Si por alguna idea peregrina nos damos a la tarea de guardar datos de ambos conceptos en un mismo documento XML debemos hallar una manera de diferenciarlos para que no haya lugar a dudas. Para ello podemos -y debemos- asignar un prefijo para eliminar cualquier duda, pero ¿qué prefijo usaremos? Pongamos por caso que los bancos -instituciones financieras- le colocamos el prefijo "**i:**" y los bancos -objetos- los prefijamos con **"o:**" (notemos que no violan la sintaxis de nombres de elementos ya que comienzan con una letra, no comienzan con "xml" ni contiene espacios):

```
                Banco de Venezuela          Banco Mercantil          Ba
nco Bicentenario                        banco de madera          banco de m
etal          taburete
```

Rapidamente notamos, si fueramos un lector ajeno que recibiera este documento XML, *¿qué diablos significa ambos prefijos, aparte de separarlos sintacticamente?* Bien podríamos colocar un comentario para hacer alusión a qué nos referimos ("i:"institución financiera, "o:"obejto), pero eso funciona solo para nosotros los humanos, los ordenadores obvian estas observaciones ¿qué otra solución podemos echar mano?

Espacios de nombres en XML: XMLNS.

Los espacios de nombres ("Name Spaces") en XML (juntos serían XMLNS) es el concepto para definir los prefijos que querramos usar en cualquier documento XML. Este "Name Spaces" son unos atributos y como tales debemos colocarlos en el elemento "padre" con un valor bajo la forma de URI (Uniform Resource Identifier) que bien puede ser a su vez un URL (Uniform Resource Locator) o un URN (Uniform Resource Name).

Para este nuestro caso vamos a usar el modelo de comentario y el modelo de URL *todo esto apuntando siempre a que sea un ser humano el que va a leer la información (y luego veremos el caso de si es un ordenador el que "procesará" la información):*

```
              Banco de Venezuela
      Banco Mercantil
      Banco Bicentenario
      banco de madera          banco de metal          taburete
```

También podemos declarar dichos atributos en el elemento raíz para darle mayor claridad a

nuestro código y es igualmente válido:

Recordemos que los documentos XML permiten los espacios en blanco para precisamente indentar para clarificar, por ello este formato de elemento raíz.

HTML DOM

Fuentes consultadas.

En idioma castellano.

En idioma inglés.

- «Document Object Model (DOM)» by W3C.
- «JavaScript HTML DOM» at w3schools.
- «HTML Parsing and Screen Scraping with the Simple HTML DOM Library».
- «HTML Parser for PHP-4» by José Solorzano.
- «PHP Simple HTML DOM Parser».
- «XML tutorial» at W3Schools.