1 Curso POO PHP DOM

1.1 Sumario

1.2 DOM

A **extensión DOM** é moito máis potente e complexa que SimpleXML. Entre outras características, permite diferenciar os tipos de nodos e traballar cos comentarios e as instrucións de procesamento dos documentos XML. Imos ver algunhas das súas principais capacidades.

1.2.1 Cargar un documento XML

Para cargar un documento XML en memoria empregando a extensión DOM, primeiro temos que instanciar un obxecto da clase **DOMDocument**, e posteriormente empregaremos un dos seguintes métodos:

- **load**. Carga un documento XML do ficheiro que se indique.

```
$xml = new DOMDocument();
$xml->load('exemplo.xml');
```

- **loadXML**. Carga un documento XML tomando como orixe unha cadea de texto.

```
$xml = new DOMDocument();
$xml->loadXML($texto_xml);
```

Moitas veces convén eliminar os nodos de texto baleiros ao importar un documento XML. Para acadalo, debemos empregar a propiedade **preserveWhiteSpace** da clase *DOMDocument* antes de procesar o arquivo ou cadea de texto.

```
$xml = new DOMDocument();
$xml->preserveWhiteSpace = false;
$xml->load('exemplo.xml');
```

Tamén é posible empregar a función **dom_import_simplexml** para converter un obxecto SimpleXML a un obxecto **DOMElement**, de xeito inverso ao que facíamos coa función *simplexml_import_dom*.

```
$sxml = simplexml_load_file('exemplo.xml');
// Creamos o novo nodo a partir do SimpleXML
$dom_element = dom_import_simplexml($sxml);
// E o importamos a un novo documento
$dxml = new DOMDocument('1.0', 'UTF-8');
$dom_node = $dxml->importNode($dom_element, true);
$dxml->appendChild($dom_node);
```

O constructor da clase DOMDocument admite dous parámetros, a versión e o xogo de caracteres do documento, o que nos permite crear novos documentos XML baleiros.

```
$xml = new DOMDocument('1.0', 'UTF-8');
```

1.2.2 Acceder ao contido dun documento XML

Ao cargar un documento XML en memoria empregando a extensión DOM, créase unha árbore de obxectos **DOMNode**. Cada un destes obxectos ten **propiedades** para:

- Acceder ao primeiro fillo (**firstChild**) e ao último fillo (**lastChild**).

- Acceder ao nodo pai (**parentNode**).
- Acceder aos nodos seguinte (**nextSibling**) e anterior (**previousSibling**) na orde do documento XML.
- Obter unha lista **DOMNodeList** dos seus nodos fillos (**childNodes**). A clase *DOMNodeList* inclúe un método **item** para acceder aos seus elementos pola súa ubicación na lista, e unha propiedade **length** que permite coñecer o número de nodos na lista.
- Obter o nome (**nodeName**) e o valor (**nodeValue**) do nodo, dependendo do seu tipo.
- Obter o tipo de nodo (**nodeType**), referido por unha das constantes que define a extensión. Algunhos dos valores posibles son XML_ELEMENT_NODE (é un nodo elemento), XML_ATTRIBUTE_NODE (nodo atributo), XML_TEXT_NODE (nodo de texto) ou XML_COMMENT_NODE (nodo comentario). Dependendo do tipo de nodo, o obxecto pertencerá a unha das clases que herdan de DOMNode: **DOMElement**, **DOMAttr**, **DOMText**, **DOMComment**, ..., cada unha das cales define os seus propios métodos e propiedades.

  Por exemplo, para acceder ao valor dun atributo emprégase a propiedade **value** da clase *DOMAttr*, para a etiqueta dun elemento temos a propiedade **tagName** da clase *DOMElement*, e para obter o texto dun nodo pódese facer coa propiedade **wholeText** da clase *DOMText*.

O seguinte script obtén o nome e valor do elemento *<importe>* presente no documento de exemplo.

```
$xml = new DOMDocument();
$xml->preserveWhiteSpace = false;
$xml->load('exemplo.xml');

$raiz = $xml->documentElement;
echo "Etiqueta: " . $raiz->firstChild->firstChild->tagName . "<br />";
echo "Valor: " . $raiz->firstChild->firstChild->firstChild->wholeText . "<br />";
```

Os obxectos da **clase DOMElement** teñen métodos para acceder aos seus atributos e elementos fillos, por exemplo:

- **hasAttribute** comproba se existe ou non no elemento un atributo co nome que indiquemos.
- **getAttribute** devolve unha cadea co valor do atributo que indiquemos. Tamén é posible devolver un nodo atributo no canto dunha cadea de texto, empregando o método **getAttributeNode**.
- **getElementsByTagName**, tamén presente na clase *DOMNode*, devolve unha lista de nodos (*DOMNodeList*) cos descendentes cuxo nome correspóndese co que indiquemos.

O seguinte script obtén os nomes dos actores no documento XML de exemplo.

```
$xml = new DOMDocument();
$xml->preserveWhiteSpace = false;
$xml->load('exemplo.xml');

$actores = $xml->getElementsByTagName('actor');
foreach ($actores as $actor) {
    $nome = $actor->getElementsByTagName('nome')->item(0);
    echo $nome->nodeValue . "<br />";
}
```

1.2.3 Modificar o contido dun documento XML

Os distintos tipos de nodos da extensión DOM inclúen métodos para engadir e eliminar nodos da árbore, como **appendChild**, **insertBefore**, **removeChild** ou **removeAttribute**.

Antes de engadir un nodo, deberemos crearlo no documento empregando os métodos **createElement**, **createAttribute** ou semellantes, dependendo do tipo de nodo.

Por exemplo, para facer no documento de exemplo cambios semellantes aos que vimos coa extensión SimpleXML, faríamos:

```
$xml = new DOMDocument();
$xml->preserveWhiteSpace = false;
$xml->load('exemplo.xml');

$pelicula = $xml->getElementsByTagName('película')->item(0);
$titulo = $pelicula->getElementsByTagName('título')->item(0);
```

```
$titulo->nodeValue = "O Santo";

$url = $xml->createElement('url','http://www.imdb.com/name/nm0000174/');
$actor = $xml->getElementsByTagName('actor')->item(1);
$actor->appendChild($url);

$version = $xml->createAttribute('versión');
$version->value = "1.0";
$xml->documentElement->appendChild($version);
```

Como no caso da extensión SimpleXML, podemos pasar o documento modificado a un ficheiro (método **save** da clase *DOMDocument*) ou a unha cadea de texto (método **saveXML**).

1.2.4 Validación de documentos XML

Empregando DOM temos acceso a varios mecanismos de validacións para os documentos XML.

- Validación **mediante DTD**. Pódese validar o documento XML a partires do seu DTD mentres se le poñendo a *true* a propiedade **validateOnParse** da clase *DOMDocument* antes de cargar o documento. Tamén é posible executar a validación unha vez cargado o documento empregando o método **validate**.

- Validación **mediante XML Schema**. A validación mediante XML Schema pode facerse indicando o nome do ficheiro (método **schemaValidate**) ou a cadea de texto (método **schemaValidateSource**) nos que se almacena o esquema.

- Validación **mediante RelaxNG**. De xeito semellante ao caso anterior, pode validarse fronte ao contido dun ficheiro (método **relaxNGValidate**) ou dunha cadea de texto (método **relaxNGValidateSource**).

1.2.5 Avaliación de expresións XPath

Na extensión DOM inclúese a clase **DOMXPath**, que permite avaliar expresións XPath sobre un documento XML almacenado nun obxecto *DOMDocument*. Ao instanciar a clase débese indicar o documento XML sobre o que se van a avaliar as expresións. Os métodos **evaluate** e **query**. A primeira delas intenta devolver un valor simple, e a segunda devolve sempre como resultado un conxunto de nodos (*DOMNodeList*).

Por exemplo, para obter os nomes das actrices presentes no documento de exemplo podemos facer:

```
$xml = new DOMDocument();
$xml->preserveWhiteSpace = false;
$xml->load('exemplo.xml');

$xpath = new DOMXPath($xml);
$consulta = '//videoteca/actor[sexo = "muller"]';
$actores = $xpath->query($consulta);

foreach ($actores as $actor) {
    $nome = $actor->getElementsByTagName('nome')->item(0);
    echo $nome->nodeValue . "<br />";
}
```

--Víctor Lourido 12:13 24 jul 2013 (CEST)