1 Curso POO PHP XMLReader e XMLWriter

1.1 Sumario

1.2 XMLReader e XMLWriter

As extensións **XMLReader** e **XMLWriter** veñen habilitadas por defecto dende a versión 5.1.2 de PHP. Permiten traballar con documentos XML empregando fluxos do mesmo xeito que se fai cos ficheiros.

1.2.1 XMLReader

Empregando XMLReader podemos validar un documento a medida que se vai procesando, e detener as tarefas de procesamento en calquera momento.

Para **abrir un documento XML**, primeiro debemos crear unha instancia da **clase XMLReader**, e empregar un dos seguintes métodos:

- **open**. Abre un documento a partir dun URI.

```
$xml = new XMLReader();
$xml->open('exemplo.xml');
```

- **XML**. Abre un documento XML almacenado nunha cadea de texto.

```
$xml = new XMLReader();
$xml->XML($texto);
```

O seguinte paso é, cando sexa preciso, utilizar o método **setParserProperty** para **establecer as opcións do procesador**. Unha das máis empregadas é **XMLReader::VALIDATE**, que habilita a validación mediante DTD.

```
$xml->setParserProperty(XMLReader::VALIDATE, true);
```

A continuación o procedemento habitual **crear un bucle** que vaia chamando ao método **read** para avanzar ao seguinte nodo, comprobando os tipos dos nodos coa propiedade **nodeType** ou os seus contidos con **value**. Cando se chega ao final do documento o método *read* devolve *false*. O método **close** pecha o documento XML orixe.

```
while ($xml->read()) {
   if ($reader->nodeType == XMLREADER::ELEMENT) print $reader->localName . "<br />";
}
```

Algunhas das constantes que se definen na clase XMLReader para os tipos de nodos son:

- *XMLReader::ELEMENT*. Comenzo dun elemento.
- *XMLReader::END_ELEMENT*. Final dun elemento.
- *XMLReader::TEXT*. Texto.
- *XMLReader::PI*. Instrucción de procesamento.
- *XMLReader::COMMENT*. Comentario.

Por exemplo, para obter a mesma táboa HTML da película que obtivemos con XMLParse, a partires da que figura no documento de exemplo, pódese facer:

```
$xml = new XMLReader();
$xml->open('exemplo.xml');

while ($xml->read()) {
switch($xml->nodeType) {
case XMLREADER::ELEMENT:
$nome_nodo = $xml->localName;
```

```
switch ($nome_nodo) {
case 'película' :
echo '<table>';
break;
case "título":
echo '<tr> <th colspan="2">';
break;
case "títuloorixinal":
echo ' (';
break;
case "director":
case "xénero":
case "duración":
echo '<tr><td>' . $nome_nodo . '</td><td>';
}
break;
case XMLReader::END_ELEMENT:
switch($xml->localName) {
case "película":
echo '</table>';
break;
case "títuloorixinal":
echo ')</th></tr>';
break;
case "director":
echo '</td></tr>';
}
$nome_nodo = "";
break;
case XMLReader::TEXT:
switch($nome_nodo) {
case "película":
case "título":
case "títuloorixinal":
case "director":
case "xénero":
case "duración":
echo $xml->value;
}
}
}

$xml->close();
```

Unha forma sinxela de percorrer os atributos dun elemento é empregando a propiedade **hasAttributes**, que indica se o elemento ten ou non atributos, e o método **moveToNextAttribute** para movernos de un en un.

```
...
case XMLREADER::ELEMENT:
   if($xml->hasAttributes)
      while($xml->moveToNextAttribute())
         echo 'Atributo ' . $xml->localName . ', valor = "' . $xml->value . '" <br />';
...
```

1.2.1.1 Validación

XMLReader pode validar o documento conforme o vai lendo. Admite tres tipos de validación.

- Mediante **DTD**. Para activar a validación por DTD, débense establecer as seguintes opcións:

```
$xml->setParserProperty(XMLReader::LOADDTD,  TRUE);
$xml->setParserProperty(XMLReader::VALIDATE, TRUE);
```

- Mediante **XML Schema**. Neste caso será necesario empregar o método **setSchema** `para indicar a ubicación do documento de validación.

```
$xml->setSchema('esquema.xsd');
```

- Mediante **Relax NG**. Temos dúas posibilidades: indicar o arquivo de validación con **setRelaxNGSchema** ou unha cadea de texto co seu contido con **setRelaxNGSchemaSource**.

```
$xml->setRelaxNGSchemaSource('esquema.rng');
```

En calquera dos casos anteriores, o método **isValid** devolverá *true* se o que se leva lido do documento é valido ou *false* en caso contrario.

```
$xml = new XMLReader();
$xml->open('exemplo.xml');
$xml->setSchema('esquema.xsd');

while ($xml->read()) {
   ...
}
if(!$xml->is_valid()) exit('Erro de validación do documento XML!');
```

1.2.2 XMLWriter

XMLWriter é a contrapartida de XMLReader. Aporta unha forma sinxela de crear documentos XML ben formados. O seu funcionamento é semellante ao de XMLReader.

Para **crear un documento XML**, primeiro debemos crear unha instancia da **clase XMLWriter**, e empregar un dos seguintes métodos:

- **openURI**. Almacena o documento XML nun URI.

```
$xml = new XMLWriter();
$xml->openURI('exemplo.xml');
```

- **openMemory**. Almacena o documento XML en memoria.

```
$xml = new XMLWriter();
$xml->openMemory();
```

  Neste caso empregando o método **outputMemory** obteremos o documento XML que levamos escrito en memoria.

Posteriormente para **crear o contido do documento**, deberemos empregar chamadas aos distintos métodos *start...*, *end...* ou *write...*.

- Para **crear un novo elemento**, podemos emprear o método **writeElement** indicando o seu nome e o seu contido

```
$xml->writeElement('xénero', 'Acción');
```

  Ou chamar primeiro a **startElement** indicando o seu nome, e despois a **endElement**. Deste xeito entre as dúas chamadas poderemos crear atributos para o elemento con **startAttribute** e **endAttribute** ou directamente con **writeAttribute**.

```
$xml->startElement('importe');
$xml->writeAttribute('moneda', 'dólar');
$xml->endElement();
```

- Do mesmo xeito tamén é posible **crear comentarios** (**startComment** con **endComment**, ou **writeComment**), **instruccións de procesamento** (**startPI** con **endPI**, ou **writePI**), etc.

- Para **crear texto** chamaremos ao método **text** indicando o seu contido

```
$xml->startElement('importe');
$xml->writeAttribute('moneda', 'dólar');
$xml->text('13.56');
$xml->endElement();
```

- Para **crear a declaración XML** emprégase o método **startDocument**, indicando opcionalmente a versión da linguaxe e o sistema de codificación empregado. Do mesmo xeito, cando remata o noso documento deberemos indicarlo chamando ao método **endDocument**.

```
$xml = new XMLWriter();
$xml->openURI('exemplo.xml');
$xml->startDocument('1.0', 'UTF-8');
...
$xml->endDocument();
```

- O método **flush** volca os búferes no disco. Debe chamarse para finalizar as labores de escritura. No caso de gardar o documento XML en memoria, devolve o seu contido.

Por exemplo, o seguinte código crea un documento XML semellante ao documento de exemplo (sen os elementos "<actor>").

```
$xml = new XMLWriter();
$xml->openURI('exemplo.xml');
$xml->setIndent(true);

$xml->startDocument('1.0', 'UTF-8');
$xml->startElement('videoteca');
$xml->writeAttribute('data_creación', '24/02/2009');
$xml->startElement('película');
$xml->writeAttribute('id', '1');
$xml->startElement('importe');
$xml->writeAttribute('moneda', 'dólar');
$xml->text('13.56');
$xml->endElement();
$xml->writeElement('título', 'El Santo');
$xml->writeElement('títuloorixinal', 'The Saint');
$xml->writeElement('ano', '1997');
$xml->writeElement('director', 'Phillip Noyce');
$xml->writeElement('xénero', 'Acción');
$xml->writeElement('duración', '111');
$xml->writeComment('  Elisabeth Shue ');
$xml->startElement('actúa');
$xml->writeAttribute('id', '51');
$xml->endElement();
$xml->writeComment('  Val Kilmer ');
$xml->startElement('actúa');
$xml->writeAttribute('id', '156');
$xml->endElement();
$xml->endElement();   // Remata o elemento "película"
$xml->endElement();   // Remata o elemento "videoteca"
$xml->endDocument();

$xml->flush();
```

O método **setIndent** habilita ou inhabilita a indentación, mellorando a lexibilidade do documento resultante.

--Víctor Lourido 01:24 25 jul 2013 (CEST)