Parsear XML & JSON

Antonio Espín Herranz

Contenidos

- DOM / SAX
- SimpleXML
- JSON

XML: DOM / SAX

Introducción

- PHP soporta el procesamiento de ficheros XML utilizando DOM y SAX.

- SAX: dirige el procesamiento del fichero XML mediante eventos, representados por funciones de php.

- DOM: Analiza el documento a partir de un árbol de nodos (representados por objetos y colecciones).

DOM

- Permite la creación y la edición de documentos XML.
- Especificación W3C.
- Proporciona soporte para el lenguaje de consulta Xpath.
- Consume mas memoria al cargar el contenido del fichero.
- Acceso a los nodos mediante objetos.

Clases DOMNode

- DOMDocument
- DOMElement
- DOMAttr
- DOMComment
- DOMDocumentType
- DOMNotation
- DOMEntity
- DOMEntityReference
- DOMProcessingInstruction
- DOMNameSpaceNode
- DOMDocumentFragment
- DOMCharacterData
- DOMText
- DOMCdataSection

Clases adicionales DOM

- DOMException:
  - Gestión de excepciones.
- DOMImplementation
- DOMNodeList:
  - Colección de nodos.
- DOMNamedNodeMap:
  - Colección de atributos.
- DOMXPath:
  - Consultas xpath.

Estructura

```xml
<?xml version="1.0" encoding="UTF-8"?>
<libros>
  <libro>
    <titulo>PHP Avanzado</titulo>
    <autor>Juan Pérez</autor>
    <anio>2023</anio>
  </libro>
  <libro>
    <titulo>XML en la práctica</titulo>
    <autor>Ana Gómez</autor>
    <anio>2022</anio>
  </libro>
</libros>
```

```
Document
└── Element: libros
    ├── Element: libro
    │   ├── Element: titulo → "PHP Avanzado"
    │   ├── Element: autor  → "Juan Pérez"
    │   └── Element: anio   → "2023"
    └── Element: libro
        ├── Element: titulo → "XML en la práctica"
        ├── Element: autor  → "Ana Gómez"
        └── Element: anio   → "2022"
```

Ejemplo navegación por nodos

```
function locateDescription($nodeset) {
   foreach ($nodeset AS $node) {
      if ($node->nodeType == XML_ELEMENT_NODE && $node->nodeName == 'description') {
         $GLOBALS['arNodeSet'][] = $node;
         return;
      }
      if ($node->hasChildNodes())  {  locateDescription($node->childNodes);  }
   }
}
// Código principal:
$dom = new DOMDocument();
$dom->load('course.xml');
$root = $dom->documentElement;

$arNodeSet = array();

if ($root->hasChildNodes())   {   locateDescription($root->childNodes); }

foreach ($arNodeSet AS $key=>$node) {     print "#$key: ".$node->nodeValue."\n";   }
```

Ejemplo – Navegación por nodos

```
/* Búsqueda por los nodos */
<?php
$dom = new DOMDocument();
$dom->load('course.xml');

$nodelist = $dom->getElementsByTagName('description');

foreach ($nodelist AS $key=>$node) {
   print "#$key: ".$node->nodeValue."\n";
}
?>
```

DOM: Creando un fichero XML

```
/* Creando un fichero XML*/
$doc = new DOMDocument();

$root = $doc->createElement("tree");
$doc->appendChild($root);

$root->setAttribute("att1", "att1 value");

$attr2 = $doc->createAttribute("att2");
$attr2->appendChild($doc->createTextNode("att2 value"));
$root->setAttributeNode($attr2);

$child = $root->appendChild($doc->createElement("child"));

$comment = $doc->createComment("My first Document");
$doc->insertBefore($comment, $root);

$pi = $doc->createProcessingInstruction("php", 'echo "Hello World!"');
$root->appendChild($pi);

$cdata = $doc->createCdataSection("special chars: & < > '");
$child->appendChild($cdata);

$doc->save("nuevo.xml");
```

El fichero resultante

```
<?xml version="1.0"?>
<!--My first Document-->
<tree att1="att1 value" att2="att2 value">
  <child><![CDATA[special chars: & < > ']]></child>
<?php echo "Hello World!"?>
</tree>
```

Simple API for XML (SAX)

- Procesamiento de ficheros XML línea a línea.
- Coste bajo en memoria.
- Una sola pasada por el fichero, después la información no está disponible.
- El parseo se dirige mediante eventos.

SAX: Source Document

xml_simple.xml

```
<?xml version='1.0'?>
<chapter xmlns:a="http://www.example.com/namespace-a"
                xmlns="http://www.example.com/default">
   <a:title>ext/xml</a:title>
   <para>
     First Paragraph
   </para>
   <a:section a:id="about">
     <title>About this Document</title>
     <para>
       <!-- this is a comment -->
       <?php echo 'Hi!  This is PHP version ' . phpversion(); ?>
     </para>
    </a:section>
</chapter>
```

SAX: Simple Example

xml/xml_simple.php

```
<?php
function startElement($parser, $elementname, $attributes) {
  print "* Start Element: $elementname \n";
  foreach ($attributes as $attname => $attvalue) {
    print "   $attname => $attvalue \n";
  }
}

function endElement($parser, $elementname) {
  print "* End Element: $elementname\n";
}

function charDataHandler($parser,$data) {
  if (trim($data) != "") print $data."\n";
}

function PIhandler ($parser, $target, $data) {
    print "PI: $target -> $data\n";
}

function DefaultHandler($parser, $data) {
    print "Default: $data\n";
}
```

SAX: Simple Example

sax.php

```
$parser = xml_parser_create();
 /* Desabilitar la conversión a Mayúsculas */
xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING, false);

xml_set_element_handler($parser,"startElement","endElement");
xml_set_character_data_handler($parser, "charDataHandler");
xml_set_processing_instruction_handler ($parser, "PIhandler");
xml_set_default_handler ($parser, "DefaultHandler");

if (($fp = fopen("xml_simple.xml", "r"))) {
   while ($data = fread($fp, 4096)) {
        xml_parse($parser, $data, feof($fp));
    }
}
?>
```

SAX: Resultados

```
* Start Element: chapter
   xmlns:a => http://www.example.com/namespace-a
   xmlns => http://www.example.com/default
* Start Element: a:title
ext/xml
* End Element: a:title
* Start Element: para

      First Paragraph
* End Element: para
* Start Element: a:section
   a:id => about
* Start Element: title
About this Document
* End Element: title
* Start Element: para
Default: <!-- this is a comment -->
PI: php -> echo 'Hi!  This is PHP version ' . phpversion();
* End Element: para
* End Element: a:section
* End Element: chapter
```

SAX: Error Handling

xml_error.php

```
<?php
/* Malformed document */
$data = "<root>";

$parser = xml_parser_create();

if(! xml_parse($parser, $data, TRUE)) {
   /* Normally die is or some other escape mechanism is also
   called*/
   printf("XML error: %s in line %d, column %d\n\n",
          xml_error_string(xml_get_error_code($parser)),
          xml_get_current_line_number($parser),
          xml_get_current_column_number($parser));
}

/* Magically you can also get a structured error */
$xmlError = libxml_get_last_error();
var_dump($xmlError);
?>
```

SAX: Error Handling

Resultados

```
XML error: Invalid document end in line 1, column 7

object(LibXMLError)#1 (6) {
  ["level"]=>
  int(3)
  ["code"]=>
  int(5)
  ["column"]=>
  int(7)
  ["message"]=>
  string(41) "Extra content at the end of the document
"
  ["file"]=>
  string(0) ""
  ["line"]=>
  int(1)
}
```

Simple XML

SimpleXML

- Facilita el acceso a documentos de XML.
- Opera con elementos y atributos.
- Soporte para XPath.
- Permite modificaciones del XML.
- A partir de PHP 5.1.3
  - Los elementos y los atributos se puede añadir con: addChild() y addAttribute().
  - El nombre de los nodos puede ser recuperado llamando a getName().

funciones

- $sxe = simplexml_load_file($url)
  - Cargar los datos XML a partir de un fichero o de una url.

- $sxe = simplexml_load_string($cadena)
  - Cargar los datos de una cadena de texto.

```
$texto="<?xml version='1.0'?>
<messages>
<message>
<topic>Trabajo</topic>
<text>Busco trabajo</text>
</message>
<message>
<topic>Re Trabajo</topic>
<text>Yo tambien</text>
</message>
</messages>";

$simple_xml = simplexml_load_string($texto);

foreach ($simple_xml->message as $message)
echo $message->text.' <br/>';
```

Lo convierte a objetos, operamos dentro de estos.

funciones

- Interoperabilidad con DOM: Permite cargar a partir de un DomDocument.

```
$dom = new DOMDocument();
$dom->loadXML('<books><book><title>Programacion en Java</title></book></books>');
if (!$dom) {
    echo 'Error al parsear el documento';
    exit();
}

$s = simplexml_import_dom($dom);

echo $s->book[0]->title;
```

funciones

- Incluye soporte para **Xpath**:

```
$simple_xml = simplexml_load_string($texto);
foreach ($simple_xml->xpath("//topic") as $dato)
    echo "$dato <br />";
```

  - Nos devuelve una colección con todos los nodos “topic”.

funciones

- Permite la modificación de datos:

```
$xml = simplexml_load_string($xmlstr);
echo $xml->movie[0]->characters->character[0]->name . "<br />";
$xml->movie[0]->characters->character[0]->name = 'Miss Coder';
echo $xml->asXML();
```

  - **asXML**: Devuelve una cadena XML basada en el objeto SimpleXML
  - asXML(ruta_fichero): Lo graba en el fichero indicado.

funciones

- Acceso a atributos:

```
$string = <<<XML
<a>
<foo name="one" game="lonely">1</foo>
</a>
XML;

$xml = simplexml_load_string($string);
foreach($xml->foo[0]->attributes() as $a => $b) {
    echo $a,'="',$b,"\"\n";
}
```

funciones

- Navegación por nodos hijo:

```
<?php
    $xml = simplexml_load_string(
    '<person>
    <child role="son">
      <child role="daughter"/>
    </child>
    <child role="daughter">
      <child role="son">
       <child role="son"/>
      </child>
    </child>
    </person>');

    foreach ($xml->children() as $second_gen) {
        echo ' The person begot a ' . $second_gen['role'];

        foreach ($second_gen->children() as $third_gen) {
            echo ' who begot a ' . $third_gen['role'] . ';';

            foreach ($third_gen->children() as $fourth_gen) {
                echo ' and that ' . $third_gen['role'] .
                    ' begot a ' . $fourth_gen['role'];
            }
        }
    }
    ?>
```

SimpleXML: Edicion de datos

```
$data = array(array('title'=>'Result 1', 'descript'=>'Res1 description'),
              array('title'=>'Result 2', 'descript'=>'description of Res2'),
              array('title'=>'Result 3', 'descript'=>'This is result 3'));

$rest = simplexml_load_string('<results num="0" />');
$rest['num'] = count($data);

foreach ($data AS $result_item) {
   $result = $rest->addChild('result');
   $result->addChild('title', $result_item['title']);
   $result->addChild('description');
   $result->description = $result_item['descript'];
}

$rest->asXML('editing_php513.xml');
```

SimpleXML: Resultados

```
<?xml version="1.0"?>
<results num="3">
    <result>
        <title>Result 1</title>
        <description>Res1 description</description>
    </result>
    <result>
        <title>Result 2</title>
        <description>description of Res2</description>
    </result>
    <result>
        <title>Result 3</title>
        <description>This is result 3</description>
    </result>
</results>
```

SimpleXML: Eliminar datos

```
<?php
$results = simplexml_load_file('editing_php513.xml');

/* Eliminar el título del primer elemento */
unset($results->result->title);

/* Borrado del 2º elemento */
unset($results->result[1]);

print $results->asXML();
?>
```

SimpleXML: Resultados

```
<?xml version="1.0"?>
<results num="3">
  <result>
    <description>Res1 description</description>
  </result>
  <result>
    <title>Result 3</title>
    <description>This is result 3</description>
  </result>
</results>
```

JSON

Generar json

- En nuestras clases podemos añadir un método como este: Utilizamos arrays asociativos.

```
public function toArray(): array {
    return [
        'nombre' => $this->nombre,
        'categoria' => $this->categoria,
        'precio' => $this->precio,
        'existencias' => $this->existencias
    ];
}
```

Generar json

```
$productos = [
    new Producto( … ),
    new Producto( … ),
    new Producto( … )
];

// Convertir a array simple
$productosArray = array_map(fn($p) => $p->toArray(), $productos);

// Codificar a JSON
$json = json_encode($productosArray, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);

// Guardar en archivo
file_put_contents(__DIR__ . "/productos.json", $json);

echo "Archivo JSON generado correctamente.";
```

Parsear json

- En nuestra clase podemos tener:

```
public static function fromArray(array $data): self {
    return new self(
        $data['nombre'],
        $data['categoria'],
        (float) $data['precio'],
        (int) $data['existencias']
    );
}
```

Parsear json

```php
<?php
require __DIR__ . '/../vendor/autoload.php';

use App\data\Producto;

// Leer el contenido del archivo JSON
$json = file_get_contents(__DIR__ . "/productos.json");

// Decodificar a array
$productosArray = json_decode($json, true);

// Convertir cada elemento en un objeto Producto
$productos = array_map(fn($item) => Producto::fromArray($item), $productosArray);

// Mostrar los productos
foreach ($productos as $producto) {
    echo $producto->__toString() . "<br>";
}
```