



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)

[Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

[Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Errors](#)

[Exceptions](#)

[Generators](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

[Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[Using Register Globals](#)

[User Submitted Data](#)

[Magic Quotes](#)

[Hiding PHP](#)

[Keeping Current](#)

[Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)

[Handling file uploads](#)

[Using remote files](#)

[Connection handling](#)

[Persistent Database Connections](#)

[Safe Mode](#)
[Command line usage](#)
[Garbage Collection](#)
[DTrace Dynamic Tracing](#)

[Function Reference](#)

[Affecting PHP's Behaviour](#)
[Audio Formats Manipulation](#)
[Authentication Services](#)
[Command Line Specific Extensions](#)
[Compression and Archive Extensions](#)
[Credit Card Processing](#)
[Cryptography Extensions](#)
[Database Extensions](#)
[Date and Time Related Extensions](#)
[File System Related Extensions](#)
[Human Language and Character Encoding Support](#)
[Image Processing and Generation](#)
[Mail Related Extensions](#)
[Mathematical Extensions](#)
[Non-Text MIME Output](#)
[Process Control Extensions](#)
[Other Basic Extensions](#)
[Other Services](#)
[Search Engine Extensions](#)
[Server Specific Extensions](#)
[Session Extensions](#)
[Text Processing](#)
[Variable and Type Related Extensions](#)
[Web Services](#)
[Windows Only Extensions](#)
[XML Manipulation](#)
[GUI Extensions](#)

Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top
g h	Goto homepage
g s	Goto search (current page)
/	Focus search box

[Tratar con los errores XML »](#)

[« Ejemplos](#)

- [Manual de PHP](#)
- [Referencia de funciones](#)
- [Manipulación de XML](#)
- [SimpleXML](#)
- [Ejemplos](#)

Change language: Spanish ▼

[Edit Report a Bug](#)

Uso básico de SimpleXML ¶

Muchos ejemplos en esta referencia requieren un string XML. En vez de repetir este string en cada ejemplo, se ha puesto en un fichero que se incluye en cada ejemplo. Este fichero se muestra en la siguiente sección de ejemplo. Alternativamente, puede crearse un documento XML y leerlo con [simplexml_load_file\(\)](#).

Ejemplo #1 El fichero ejemplo.php a incluir con el string XML

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<peliculas>
  <pelicula>
    <titulo>PHP: Tras el Analilizador</titulo>
    <personajes>
      <personaje>
        <nombre>Srta. Programadora</nombre>
        <actor>Onlivia Actora</actor>
      </personaje>
      <personaje>
        <nombre>Sr. Programador</nombre>
        <actor>El Act&#211;r</actor>
      </personaje>
    </personajes>
    <argumento>
      Así que, este lenguaje. Es como, un lenguaje de programación. ¿0 es un
      lenguaje de script? Lo descubrirás en esta intrigante y temible parodia
      de un documental.
    </argumento>
    <grandes-frases>
      <frase>PHP soluciona todos los problemas web</frase>
    </grandes-frases>
    <puntuacion tipo="votos">7</puntuacion>
    <puntuacion tipo="estrellas">5</puntuacion>
  </pelicula>
</peliculas>
XML;
?>
```

La simplicidad de SimpleXML se ve claramente cuando se extrae un string o un número de un documento XML básico.

Ejemplo #2 Obtener <argumento>

```
<?php
include 'ejemplo.php';
```

```
$peliculas = new SimpleXMLElement($xmlstr);

echo $peliculas->pelicula[0]->argumento;
?>
```

El resultado del ejemplo sería:

Así que, este lenguaje. Es como, un lenguaje de programación. ¿O es un lenguaje de script? Lo descubrirás en esta intrigante y temible parodia de un documental.

El acceso a elementos dentro de un documento XML que contiene caracteres no permitidos por la convención de nombres de PHP (p.ej., el guión) puede realizarse encapsulando el nombre del elemento dentro de un par de llaves y comillas simples.

Ejemplo #3 Obtener *<frase>*

```
<?php
include 'ejemplo.php';

$peliculas = new SimpleXMLElement($xmlstr);

echo $peliculas->pelicula->{'grandes-frases'}->frase;
?>
```

El resultado del ejemplo sería:

PHP soluciona todos los problemas web

Ejemplo #4 Acceder a elementos no únicos en SimpleXML

Cuando existen múltiples instancias de un elemento como hijos de un único elemento padre, se aplican las técnicas normales de iteración.

```
<?php
include 'ejemplo.php';

$peliculas = new SimpleXMLElement($xmlstr);

/* Para cada <personaje>, se muestra cada <nombre>. */
foreach ($peliculas->pelicula->personajes->personaje as $personaje) {
    echo $personaje->nombre, ' interpretado por ', $personaje->actor, PHP_EOL;
}

?>
```

El resultado del ejemplo sería:

Srta. Programadora interpretado por Onlivia Actora
Sr. Programador interpretado por El Actór

Nota:

Las propiedades (*\$peliculas->pelicula* en el ejemplo anterior) no son arrays. Son objetos [iterables](#) y [accesibles](#).

Ejemplo #5 Utilizar atributos

Hasta aquí, únicamente se ha cubierto el trabajo de leer nombres de elementos y sus valores. SimpleXML puede también acceder a los atributos de los elementos. Para acceder a ellos, se realiza como si fuesen elementos de un [array](#).

```
<?php
include 'ejemplo.php';

$peliculas = new SimpleXMLElement($xmlstr);

/* Acceder a los nodos <puntuacion> de la primera película.
 * Mostrar la escala de puntuación también. */
foreach ($peliculas->pelicula[0]->puntuacion as $puntuacion) {
    switch((string) $puntuacion['tipo']) { // Obtener los atributos como índices del elemento
        case 'votos':
            echo $puntuacion, ' votos positivos';
            break;
        case 'estrellas':
            echo $puntuacion, ' estrellas';
            break;
    }
}
?>
```

El resultado del ejemplo sería:

7 votos positivos5 estrellas

Ejemplo #6 Comparar elementos y atributos con texto

Para comparar un elemento o atributo con un string o pasarlo a una función que requiera un string, debe realizarse una conversión a string empleando *(string)*. De lo contrario, PHP trata al elemento como un objeto.

```
<?php
include 'ejemplo.php';

$peliculas = new SimpleXMLElement($xmlstr);

if ((string) $peliculas->pelicula->titulo == 'PHP: Tras el Analilzador') {
    print 'Mi película favorita.';
}

echo htmlentities((string) $peliculas->pelicula->titulo);
?>
```

El resultado del ejemplo sería:

Mi película favorita.PHP: Tras el Analilzador

Ejemplo #7 Comparar dos elementos

Dos SimpleXMLElements son considerados distintos incluso cuando ambos apuntan al mismo elemento desde PHP 5.2.0.

```
<?php
include 'ejemplo.php';

$pelicula1 = new SimpleXMLElement($xmlstr);
$pelicula2 = new SimpleXMLElement($xmlstr);
```

```
var_dump($pelicula1 == $pelicula2); // falso desde PHP 5.2.0
?>
```

El resultado del ejemplo sería:

```
bool(false)
```

Ejemplo #8 Utilizar XPath

SimpleXML incorpora soporte para XPath. Para encontrar todos los elementos *<personaje>*:

```
<?php
include 'ejemplo.php';

$películas = new SimpleXMLElement($xmlstr);

foreach ($películas->xpath('//personaje') as $personaje) {
    echo $personaje->nombre . ' interpretado por ' . $personaje->actor, PHP_EOL;
}
?>
```

'/' actúa como un comodín. Para especificar una ruta absoluta, hay que omitir una de las dos barras.

El resultado del ejemplo sería:

```
Srta. Programadora interpretado por Onlivia Actora
Sr. Programador interpretado por El Actór
```

Ejemplo #9 Establecer valores

Los datos en SimpleXML no tienen que ser constantes. El objeto permite que se manipulen todos sus elementos.

```
<?php
include 'ejemplo.php';
$películas = new SimpleXMLElement($xmlstr);

$películas->película[0]->personajes->personaje[0]->nombre = 'Srta. Programadora';

echo $películas->asXML();
?>
```

El resultado del ejemplo sería:

```
<?xml version="1.0" standalone="yes"?>
<películas>
  <película>
    <titulo>PHP: Tras el Analilzador</titulo>
    <personajes>
      <personaje>
        <nombre>Srta. Programadora</nombre>
        <actor>Onlivia Actora</actor>
      </personaje>
      <personaje>
        <nombre>Sr. Programador</nombre>
        <actor>El Act&#xD3;r</actor>
      </personaje>
    </personajes>
    <argumento>
      Así que, este lenguaje. Es como, un lenguaje de programación. ¿0 es un
      lenguaje de script? Lo descubrirás en esta intrigante y temible parodia
      de un documental.
    </argumento>
```

```

<grandes-frases>
  <frase>PHP soluciona todos los problemas web</frase>
</grandes-frases>
<puntuacion tipo="votos">7</puntuacion>
<puntuacion tipo="estrellas">5</puntuacion>
</pelicula>
</peliculas>
+

```

Ejemplo #10 Añadir elementos y atributos

Desde PHP 5.1.3, SimpleXML tiene la capacidad de añadir fácilmente hijos y atributos.

```

<?php
include 'ejemplo.php';
$peliculas = new SimpleXMLElement($xmlstr);

$personaje = $peliculas->pelicula[0]->personajes->addChild('personaje');
$personaje->addChild('nombre', 'Sr. Analizador');
$personaje->addChild('actor', 'John Doe');

$puntuacion = $peliculas->pelicula[0]->addChild('puntuacion', 'Todos los públicos');
$puntuacion->addAttribute('tipo', 'clasificacion');

echo $peliculas->asXML();
?>

```

El resultado del ejemplo sería:

```

<?xml version="1.0" standalone="yes"?>
<peliculas>
  <pelicula>
    <titulo>PHP: Tras el Analizador</titulo>
    <personajes>
      <personaje>
        <nombre>Srta. Programadora</nombre>
        <actor>Onlivia Actora</actor>
      </personaje>
      <personaje>
        <nombre>Sr. Programador</nombre>
        <actor>El Actor</actor>
      </personaje>
      <personaje><nombre>Sr. Analizador</nombre><actor>John Doe</actor></personaje></personajes>
    <argumento>
      Así que, este lenguaje. Es como, un lenguaje de programación. ¿O es un
      lenguaje de script? Lo descubrirás en esta intrigante y temible parodia
      de un documental.
    </argumento>
    <grandes-frases>
      <frase>PHP soluciona todos los problemas web</frase>
    </grandes-frases>
    <puntuacion tipo="votos">7</puntuacion>
    <puntuacion tipo="estrellas">5</puntuacion>
    <puntuacion tipo="clasificacion">Todos los públicos</puntuacion>
  </pelicula>
</peliculas>

```

Ejemplo #11 Interoperatibilidad con DOM

PHP tiene un mecanismo para convertir nodos XML entre los formatos SimpleXML y DOM. Este ejemplo muestra cómo cambiar un elemento DOM a SimpleXML.

```

<?php
$dom = new DOMDocument;

```

```

$dom->loadXML('<libros><libro><titulo>bla</titulo></libro></libros>');
if (!$dom) {
    echo 'Error al analizar el documento';
    exit;
}

$s = simplexml_import_dom($dom);

echo $s->libro[0]->titulo;
?>

```

El resultado del ejemplo sería:

bla

[+ add a note](#)

User Contributed Notes 12 notes

[up](#)
[down](#)

49

[jishcem at gmail dot com ¶](#)

3 years ago

For me it was easier to use arrays than objects,

So, I used this code,

```

$xml = simplexml_load_file('xml_file.xml');

$json_string = json_encode($xml);

$result_array = json_decode($json_string, TRUE);

```

Hope it would help someone

[up](#)
[down](#)

17

[rowan dot collins at gmail dot com ¶](#)

1 year ago

There is a common "trick" often proposed to convert a SimpleXML object to an array, by running it through `json_encode()` and then `json_decode()`. I'd like to explain why this is a bad idea.

Most simply, because the whole point of SimpleXML is to be easier to use and more powerful than a plain array. For instance, you can write `<?php $foo->bar->baz['bing'] ?>` and it means the same thing as `<?php $foo->bar[0]->baz[0]['bing'] ?>`, regardless of how many bar or baz elements there are in the XML; and if you write `<?php (string)$foo->bar[0]->baz[0] ?>` you get all the string content of that node - including CDATA sections - regardless of whether it also has child elements or attributes. You also have access to namespace information, the ability to make simple edits to the XML, and even the ability to "import" into a DOM object, for much more powerful manipulation. All of this is lost by turning the object into an array rather than reading understanding the examples on this page.

Additionally, because it is not designed for this purpose, the conversion to JSON and back will actually lose information in some situations. For instance, any elements or attributes in a namespace will simply be discarded, and any text content will be discarded if an element also has children or attributes. Sometimes, this won't matter, but if you get in the habit of converting everything to arrays, it's going to sting you eventually.

Of course, you could write a smarter conversion, which didn't have these limitations, but at that point, you are getting no value out of SimpleXML at all, and should just use the lower level XML Parser functions, or the XMLReader class, to create your structure. You still won't have the extra convenience functionality of SimpleXML, but that's your loss.

[up](#)
[down](#)

14

[ie dot raymond at gmail dot com ¶](#)

6 years ago

If you need to output valid xml in your response, don't forget to set your header content type to xml in addition to echoing out the result of asXML():

```
<?php
```

```
$xml=simplexml_load_file('...');
```

```
...
```

```
...xml stuff
```

```
...
```

```
//output xml in your response:
```

```
header('Content-Type: text/xml');
```

```
echo $xml->asXML();
```

```
?>
```

[up](#)
[down](#)

6

[kdos ¶](#)

5 years ago

Using stuff like: `is_object($xml->module->admin)` to check if there actually is a node called "admin", doesn't seem to work as expected, since simplexml always returns an object- in that case an empty one - even if a particular node does not exist.

For me good old `empty()` function seems to work just fine in such cases.

Cheers

[up](#)
[down](#)

3

[Max K. ¶](#)

6 years ago

From the README file:

SimpleXML is meant to be an easy way to access XML data.

SimpleXML objects follow four basic rules:

- 1) properties denote element iterators
- 2) numeric indices denote elements
- 3) non numeric indices denote attributes
- 4) string conversion allows to access TEXT data

When iterating properties then the extension always iterates over all nodes with that element name. Thus method `children()` must be called to iterate over subnodes. But also doing the following:

```
foreach ($obj->node_name as $elem) {  
    // do something with $elem  
}
```

always results in iteration of 'node_name' elements. So no further

check is needed to distinguish the number of nodes of that type.

When an elements TEXT data is being accessed through a property then the result does not include the TEXT data of subelements.

Known issues

=====

Due to engine problems it is currently not possible to access a subelement by index 0: \$object->property[0].

[up](#)

[down](#)

4

[gkokmdam at zonnet dot nl ¶](#)

5 years ago

A quick tip on xpath queries and default namespaces. It looks like the XML-system behind SimpleXML has the same workings as I believe the XML-system .NET uses: when one needs to address something in the default namespace, one will have to declare the namespace using registerXPathNamespace and then use its prefix to address the otherwise in the default namespace living element.

```
<?php
```

```
$string = <<<XML
```

```
<?xml version='1.0'?>
```

```
<document xmlns="http://www.w3.org/2005/Atom">
```

```
<title>Forty What?</title>
```

```
<from>Joe</from>
```

```
<to>Jane</to>
```

```
<body>
```

```
    I know that's the answer -- but what's the question?
```

```
</body>
```

```
</document>
```

```
XML;
```

```
$xml = simplexml_load_string($string);
```

```
$xml->registerXPathNamespace("def", "http://www.w3.org/2005/Atom");
```

```
$nodes = $xml->xpath("//def:document/def:title");
```

```
?>
```

[up](#)

[down](#)

2

[RiKdnUA at mail dot ru ¶](#)

3 years ago

Если кодировка XML-документа отличается от UTF-8, объявление кодировки должно следовать сразу после version='...' и перед standalone='...'. Это требование стандарта XML.

If encoding XML-document differs from UTF-8. Encoding declaration should follow immediately after the version = '...' and before standalone = '...'. This requirement is standard XML.

```
<?xml version='1.0' encoding='windows-1251' standalone='yes'?>
```

```
Ok
```

```
<?xml version='1.0' standalone='yes' encoding='windows-1251'?>
```

```
<body>Русский язык. Russian language</body>
```

```
Fatal error: Uncaught exception 'Exception' with message 'String could not be parsed as XML' in...
```

[up](#)

[down](#)

2

[bjorn at xOmail dot eu ¶](mailto:bjorn.at.xOmail.dot.eu)

7 years ago

If you're not sure the XML will be valid you'd better use:

```
<?php
$xmlObject = simplexml_load_string($xml);
// or
$xmlObject = simplexml_load_file(xml);
?>
```

Both of these return a SimpleXMLElement Object or a libXML_Error Object.

[up](#)

[down](#)

1

[php at keith tyler dot com ¶](mailto:php.at.keith.tyler.dot.com)

6 years ago

[Editor's Note: The SimpleXMLIterator class, however, does implement these methods.]

While SimpleXMLElement claims to be iterable, it does not seem to implement the standard Iterator interface functions like ::next and ::reset properly. Therefore while foreach() works, functions like next(), current(), or each() don't seem to work as you would expect -- the pointer never seems to move or keeps getting reset.

[up](#)

[down](#)

-3

[radams at circlepix com ¶](mailto:radams.at.circlepix.com)

7 years ago

To test whether an element exists:

```
<?php

$xml = <<<EOT
<?xml version='1.0' standalone='yes'?>
<root>
    <test1></test1>
    <test2 />
    <test4> </test4>
</root>
EOT;

$xmlDoc = new SimpleXMLElement($xml);

echo "Test1: \n";
var_dump($xmlDoc->test1);
echo "\n(" . (bool)$xmlDoc->test1 . ")";
echo "\n\n";

echo "Test2: \n";
var_dump($xmlDoc->test2);
echo "\n(" . (bool)$xmlDoc->test2 . ")";
echo "\n\n";

echo "Test3: \n";
var_dump($xmlDoc->test3);
echo "\n(" . (bool)$xmlDoc->test3 . ")";
echo "\n\n";

echo "Test4: \n";
```

```
var_dump($xmlDoc->test4);
echo "\n(" . (bool)$xmlDoc->test4 . ")";
echo "\n\n";
```

?>

The var_dumps for test1, test2, and test3 are identical, but the (bool) test gives a '1' for test1 and test2, and a '' for test3.

[up](#)
[down](#)

-9

[mymymymy at mymymymy dot com ¶](#)

1 year ago

```
$children = $xml->children();
var_dump($children[0]["@attributes"]);// "NULL"
var_dump($children[0]); //(2) ["@attributes"]=>array(1) {['x']=> string(1) '2'}
```

no faithfully (not true)

example:

SimpleXMLElement Object

```
(
    [@attributes] => Array
        (
            [name] => USA
        )
)
```

```
print_r($xml->country->attributes()->name);
```

show:

SimpleXMLElement Object

```
(
    [0] => USA
)
```

```
print_r((string)$xml->country->attributes()->name);
```

show:

USA

[up](#)
[down](#)

-9

[eng dot emad_2010 at yahoo dot com ¶](#)

3 years ago

Example # DOM Interoperability التشغيل البيني

PHP has a mechanism to convert XML nodes between SimpleXML and DOM formats. This example shows how one might change a DOM element to SimpleXML .

The simplexml_import_dom() function returns a SimpleXMLElement object from a DOM node.


DOM object

```
<?php
//create DOM document $dom that contain XML
$dom = new DOMDocument;
$dom->loadXML('<books><book><title>blah</title></book></books>');
if (!$dom) {
    echo 'Error while parsing the document';
    exit;
}
```

```
//convert DOM document $dom to object $books  
$books = simplexml_import_dom($dom);
```

```
//access object $books  
echo $books->book[0]->title;
```

```
?>
```

 [add a note](#)

- [Ejemplos](#)
 - [Uso básico de SimpleXML](#)
 - [Tratar con los errores XML](#)
- [Copyright © 2001-2016 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Mirror sites](#)
- [Privacy policy](#)