PHP function operators

How template.ini [PHP] PHPOperatorList turns one-argument PHP functions into template operators through eZTemplatePHPOperator, the operators it gives on a development installation (urlencode, urldecode, rawurlencode...

Source file: `doc/eng-us/help/template-operators/php-function-operators.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/template-operators/php-function-operators.md

The quickest way to give templates a PHP function is a line in `template.ini`:

INI

```ini
[PHP]
PHPOperatorList[<operator name>]=<PHP function>
```

Every entry becomes an operator implemented by `eZTemplatePHPOperator` (`lib/eztemplate/classes/eztemplatephpoperator.php`). The kernel registers them in `kernel/common/eztemplateautoload.php`: `eZPHPOperatorNameInit()` reads the names from the setting and `eZPHPOperatorInit()` creates one `eZTemplatePHPOperator` with the whole list.

Contents

1. How an entry behaves
2. The registered entries
3. The operators
4. Order and conflicts
5. Risks
6. Related pages

How an entry behaves

`eZTemplatePHPOperator::modify()` calls the PHP function with **one** argument: the piped input, or, when nothing is piped, the first parameter. Further parameters are ignored. A `null` value is passed as an empty string. The result is returned as it is. The compiled form (`phpOperatorTransformation()`) generates the same single call.

The comment in `settings/template.ini` recommends giving the operator a name of your own rather than the PHP name, so the function behind it can change later without template changes, and states the limits: one parameter, and the function must return its result by value.

The registered entries

The kernel's `settings/template.ini` lists four. The extensions active on a development installation add more through their `settings/template.ini.append.php`:

Operator	PHP function	Declared in
`urlencode`	`urlencode()`	`settings/template.ini` (also `extension/eztags`)

Operator	PHP function	Declared in
urldecode	urldecode()	settings/template.ini (also extension/eztags)
rawurlencode	rawurlencode()	settings/template.ini
rawurldecode	rawurldecode()	settings/template.ini (also extension/ezwebin)
getenv	getenv()	extension/bcgooglesitemaps
serialize	serialize()	extension/swark
unserialize	unserialize()	extension/swark
nl2br	nl2br()	extension/cjw_newsletter
strip_tags	strip_tags()	extension/ezmultiupload

The extension ezfind adds `array_merge_recursive` and `array_keys` in its settings, but it is not an active extension on a development installation, so those two are not registered here. INI arrays are merged, so the same entry in two places is harmless.

The operators

urlencode, urldecode, rawurlencode, rawurldecode

The PHP URL encoders. `urlencode` turns a space into `+` (for query strings); `rawurlencode` follows RFC 3986 and turns it into `%20` (for path segments). Input: the string, piped or as the parameter. Returns a string. Generated references: urlencode, urldecode, rawurlencode, rawurldecode.

Template

```
{def $q = 'fit & healthy recipes'}
urlencode = {$q|urlencode}
rawurlencode = {$q|rawurlencode}
urldecode( 'fit+%26+healthy' ) = {'fit+%26+healthy'|urldecode}
rawurldecode( 'fit%20%26%20healthy' ) = {'fit%20%26%20healthy'|rawurldecode}
call form: {urlencode( 'a b' )}
```

Output:

```
urlencode = fit+%26+healthy+recipes
rawurlencode = fit%20%26%20healthy%20recipes
urldecode( 'fit+%26+healthy' ) = fit & healthy
rawurldecode( 'fit%20%26%20healthy' ) = fit & healthy
call form: a+b
```

nl2br

The library has its own `nl2br` operator, `eZTemplateNl2BrOperator` (`lib/eztemplate/classes/eztemplatenl2broperator.php`), which also works element by element on an array. On a development installation the `PHPOperatorList` entry from cjw_newsletter replaces it, because `kernel/common/` is loaded after `lib/eztemplate/classes/` in `[TemplateSettings] AutoloadPathList` of `site.ini`, and a later registration of a name wins. For a string both give the same result: PHP `nl2br()`, which does not escape, so wash first (`$text|wash|nl2br`) or use the string operator break, which does the same. Generated reference: nl2br.

Template

```
{"first line
second line"|nl2br}
```

Output:

```
first line<br />
second line
```

serialize and unserialize

PHP `serialize()` and `unserialize()`. Returns a string, or the restored value. Attribute access works on the result of an operator, as `$s|unserialize.tags` shows. These two come from an extension setting, so the generated reference, which reads only the kernel settings, has no page for them.

Template

```
{def $s = hash( 'node_id', 89, 'tags', array( 'fit', 'healthy' ) )|serialize}
serialize = {$s}
unserialize = {$s|unserialize|get_type}, tags: {$s|unserialize.tags|implode( ', ' )}
```

Output:

```
serialize = a:2:{s:7:"node_id";i:89;s:4:"tags";a:2:{i:0;s:3:"fit";i:1;s:7:"healthy";}}
unserialize = array[2], tags: fit, healthy
```

getenv

PHP `getenv( name )`: the value of an environment variable of the PHP process, or `false`. The example asks for a variable that does not exist, because the environment of a server process is not something to print in documentation. It comes from an extension setting and has no generated page.

Template

```
'NO_SUCH_VARIABLE_C11'|getenv = [{'NO_SUCH_VARIABLE_C11'|getenv|choose( 'false', 'true' )}]
```

Output:

```
'NO_SUCH_VARIABLE_C11'|getenv = [false]
```

strip_tags

PHP `strip_tags()` with one argument. The same name is also an operator of `eZTemplateStringsOperator` with an allowed-tags parameter, and which one answers depends on which class the request loaded last. See the registration trap.

Order and conflicts

A name registered later replaces an earlier one. Operators from `AutoloadPathList` are registered first (library, then kernel, then `lib/ezpdf` and `kernel/private`), then those of the extensions in `ExtensionAutoloadPath`. Classes are loaded lazily, at the first use of one of their names, and loading a class registers **all** of its names again. So a `PHPOperatorList` entry that

reuses the name of a class operator is fragile, as the `strip_tags` case shows. Use a name of your own.

Risks

Every template, including those an editor can change through the template editor in the administration interface, can call these operators with any value. Do not add functions that read or change things outside their argument: `getenv` gives templates the server environment, `unserialize` of a value that comes from a request or content can create objects, and a function such as `file_get_contents` would let a template read any file the web server can read. The owsimpleoperator extension makes the same point in its settings (owsimpleoperator).

Related pages

- Registration order and overrides
- Registering operators
- PHP string functions: transform
- owsimpleoperator: PHP functions with several parameters
- All template operators (generated)
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.