PHP-Funktionsoperatoren

Wie template.ini [PHP] PHPOperatorList PHP-Funktionen mit einem Argument über eZTemplatePHPOperator zu Template-Operatoren macht, welche Operatoren sich daraus auf einer Entwicklungsinstallation ergeben (urlencode...

Source file: `doc/ger-de/hilfe/template-operatoren/php-funktionsoperatoren.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/template-operators/php-function-operators.md

Der schnellste Weg, Templates eine PHP-Funktion zur Verfügung zu stellen, ist eine Zeile in `template.ini`:

INI

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

Jeder Eintrag wird zu einem Operator, den `eZTemplatePHPOperator` (`lib/eztemplate/classes/eztemplatephpoperator.php`) implementiert. Der Kernel registriert sie in `kernel/common/eztemplateautoload.php`: `eZPHPOperatorNameInit()` liest die Namen aus der Einstellung, und `eZPHPOperatorInit()` erzeugt ein `eZTemplatePHPOperator` mit der ganzen Liste.

Inhalt

1. Wie sich ein Eintrag verhält
2. Die registrierten Einträge
3. Die Operatoren
4. Reihenfolge und Konflikte
5. Risiken
6. Verwandte Seiten

Wie sich ein Eintrag verhält

`eZTemplatePHPOperator::modify()` ruft die PHP-Funktion mit **einem** Argument auf: der gepipten Eingabe oder, wenn nichts gepipt wird, dem ersten Parameter. Weitere Parameter werden ignoriert. Ein Wert `null` wird als leere Zeichenkette übergeben. Das Ergebnis wird unverändert zurückgegeben. Die kompilierte Form (`phpOperatorTransformation()`) erzeugt denselben einzelnen Aufruf.

Der Kommentar in `settings/template.ini` empfiehlt, dem Operator einen eigenen Namen statt des PHP-Namens zu geben, damit sich die Funktion dahinter später ohne Änderungen an den Templates austauschen lässt, und nennt die Grenzen: ein Parameter, und die Funktion muss ihr Ergebnis als Wert zurückgeben.

Die registrierten Einträge

Die `settings/template.ini` des Kernels führt vier auf. Die auf einer Entwicklungsinstallation aktiven Erweiterungen fügen über ihre `settings/template.ini.append.php` weitere hinzu:

Operator	PHP-Funktion	Deklariert in
urlencode	urlencode()	settings/template.ini (auch extension/eztags)
urldecode	urldecode()	settings/template.ini (auch extension/eztags)
rawurlencode	rawurlencode()	settings/template.ini
rawurldecode	rawurldecode()	settings/template.ini (auch 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

Die Erweiterung ezfind fügt in ihren Einstellungen `array_merge_recursive` und `array_keys` hinzu, ist auf einer Entwicklungsinstallation aber keine aktive Erweiterung, daher sind diese beiden hier nicht registriert. INI-Arrays werden zusammengeführt, daher schadet derselbe Eintrag an zwei Stellen nicht.

Die Operatoren

urlencode, urldecode, rawurlencode, rawurldecode

Die URL-Kodierer von PHP. `urlencode` macht aus einem Leerzeichen ein `+` (für Query-Strings); `rawurlencode` folgt RFC 3986 und macht daraus `%20` (für Pfadsegmente). Eingabe: die Zeichenkette, gepipt oder als Parameter. Gibt eine Zeichenkette zurück. Generierte Referenzen: 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' )}
```

Ausgabe:

```
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

Die Bibliothek hat einen eigenen Operator `nl2br`, `eZTemplateNl2BrOperator` (`lib/eztemplate/classes/eztemplatenl2broperator.php`), der auch elementweise auf einem Array arbeitet. Auf einer Entwicklungsinstallation ersetzt ihn der Eintrag in `PHPOperatorList` aus cjw_newsletter, weil `kernel/common/` in `[TemplateSettings] AutoloadPathList` von `site.ini` nach `lib/eztemplate/classes/` geladen wird und eine spätere Registrierung eines Namens gewinnt. Für eine Zeichenkette liefern beide dasselbe Ergebnis: PHP `nl2br()`, das nicht maskiert; wenden Sie also zuerst wash an (`$text|wash|nl2br`) oder verwenden Sie den Zeichenketten-Operator break, der dasselbe tut. Generierte Referenz: nl2br.

Template

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

Ausgabe:

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

serialize und unserialize

PHP `serialize()` und `unserialize()`. Gibt eine Zeichenkette zurück bzw. den wiederhergestellten Wert. Attributzugriff funktioniert auf dem Ergebnis eines Operators, wie `$s|unserialize.tags` zeigt. Diese beiden kommen aus einer Einstellung einer Erweiterung, daher hat die generierte Referenz, die nur die Einstellungen des Kernels liest, keine Seite für sie.

Template

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

Ausgabe:

```
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 )`: der Wert einer Umgebungsvariablen des PHP-Prozesses oder `false`. Das Beispiel fragt nach einer Variablen, die nicht existiert, denn die Umgebung eines Serverprozesses gehört nicht in eine Dokumentation. Der Operator kommt aus einer Einstellung einer Erweiterung und hat keine generierte Seite.

Template

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

Ausgabe:

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

strip_tags

PHP `strip_tags()` mit einem Argument. Derselbe Name ist auch ein Operator von `eZTemplateStringsOperator` mit einem Parameter für erlaubte Tags, und welcher antwortet, hängt davon ab, welche Klasse die Anfrage zuletzt geladen hat. Siehe die Falle bei der Registrierung.

Reihenfolge und Konflikte

Ein später registrierter Name ersetzt einen früheren. Operatoren aus `AutoloadPathList` werden zuerst registriert (Bibliothek, dann Kernel, dann `lib/ezpdf` und `kernel/private`), dann die der Erweiterungen in `ExtensionAutoloadPath`. Klassen werden verzögert geladen, bei der ersten Verwendung eines ihrer Namen, und das Laden einer Klasse registriert **alle** ihre Namen erneut. Ein Eintrag in `PHPOperatorList`, der den Namen eines Klassenoperators wiederverwendet, ist daher fragil, wie der Fall `strip_tags` zeigt. Verwenden Sie einen eigenen Namen.

Risiken

Jedes Template, auch eines, das ein Redakteur über den Template-Editor in der Administrationsoberfläche ändern kann, kann diese Operatoren mit jedem Wert aufrufen. Fügen Sie keine Funktionen hinzu, die etwas außerhalb ihres Arguments lesen oder ändern: `getenv` gibt Templates die Serverumgebung, `unserialize` auf einen Wert aus einer Anfrage oder aus Inhalten kann Objekte erzeugen, und eine Funktion wie `file_get_contents` würde ein Template jede Datei lesen lassen, die der Webserver lesen kann. Die Erweiterung owsimpleoperator sagt in ihren Einstellungen dasselbe (owsimpleoperator).

Verwandte Seiten

- Registrierungsreihenfolge und Überschreibungen
- Operatoren registrieren
- PHP-Zeichenkettenfunktionen: umwandeln
- owsimpleoperator: PHP-Funktionen mit mehreren Parametern
- Alle Template-Operatoren (generiert)
- Freie Dokumentation: Diese Seite darf unter der GNU FDL 1.3 frei kopiert, übersetzt und verbessert werden.

> **Werden Sie Exponential-Partner**
>
> Partner teilen ihre Informationen, stellen Mittel bereit, geben Beiträge zurück und entwickeln die Produkte mit uns, und so wächst Exponential. Eine Partner-Website und ein Dashboard sind für Mitte 2027 geplant: Werden Sie noch heute Partner, damit wir schneller wachsen und neue Funktionen früher liefern.
>
> Partner: hier beginnen | Wie Sie sich bewerben

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.