PHP-Zeichenkettenfunktionen: umwandeln

Die umwandelnden Operatoren von eZTemplateStringsOperator (substr, substr_replace, ristring, rstring, strtr, str_split, str_word_count, str_getcsv, strtok, ltrim, rtrim, lcfirst, chunk_split, addslashes, stripslashes...

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

Diese Seite behandelt die Operatoren von `eZTemplateStringsOperator` (`lib/eztemplate/classes/eztemplatestringsoperator.php`), die Zeichenketten ausschneiden, ersetzen, aufteilen und maskieren. Die Abbildungsregel ist für alle gleich: Die gepipte Eingabe ist die Hauptzeichenkette der PHP-Funktion, die Template-Parameter sind ihre übrigen Argumente in ihrer Reihenfolge (wie die Funktionen abgebildet werden).

Diese Funktionen arbeiten auf **Bytes**. Für Text mit Akzenten oder anderen Multibyte-Zeichen arbeiten die Array-Operatoren `extract`, `insert`, `remove` und `replace` auf Zeichen (Array-Operatoren).

Inhalt

1. Ausschneiden: substr und substr_replace
2. Ersetzen: ristring, rstring und strtr
3. Aufteilen: str_split, str_word_count, str_getcsv und strtok
4. Ränder und Schreibung: ltrim, rtrim, lcfirst und chunk_split
5. Backslashes: addslashes, stripslashes, addcslashes, stripcslashes und quotemeta
6. strip_tags und die Registrierungsfalle
7. str_shuffle und hebrev
8. str_increment und str_decrement
9. Verwandte Seiten

Ausschneiden: substr und substr_replace

substr

PHP `substr()`. Ohne Startwert protokolliert es "Missing required parameter: start".

Parameter	Typ	Erforderlich	Standardwert
start	integer (negativ zählt vom Ende)	ja	keiner
length	integer	nein	bis zum Ende

Gibt eine Zeichenkette zurück. Generierte Referenz: substr.

substr_replace

PHP `substr_replace()`: ersetzt den Teil ab start (über length Bytes) durch den Ersatz.

Parameter	Typ	Erforderlich	Standardwert
replace	string	ja	keiner
start	integer	ja	keiner
length	integer	nein	bis zum Ende

Gibt eine Zeichenkette zurück. Generierte Referenz: substr_replace.

Template

```
{def $s = 'Healthy eating'}
substr( 8 ) = {$s|substr( 8 )}
substr( 0, 7 ) = {$s|substr( 0, 7 )}
substr( -6 ) = {$s|substr( -6 )}
substr_replace( 'Clean', 0, 7 ) = {$s|substr_replace( 'Clean', 0, 7 )}
substr_replace( '!', 14 ) = {$s|substr_replace( '!', 14 )}
```

Ausgabe:

```
substr( 8 ) = eating
substr( 0, 7 ) = Healthy
substr( -6 ) = eating
substr_replace( 'Clean', 0, 7 ) = Clean eating
substr_replace( '!', 14 ) = Healthy eating!
```

Ersetzen: ristring, rstring und strtr

ristring

PHP `str_replace( search, replace, input )`: Ersetzung **mit Beachtung der Groß- und Kleinschreibung**, trotz des "i" im Namen. Suche und Ersatz dürfen wie in PHP Arrays sein.

Parameter	Typ	Erforderlich	Standardwert
search	string oder array	ja	keiner
replace	string oder array	ja	keiner

Gibt eine Zeichenkette zurück. Generierte Referenz: ristring.

rstring

PHP `str_ireplace()`: die Ersetzung **ohne Beachtung der Groß- und Kleinschreibung**. Dieselben Parameter wie `ristring`. Generierte Referenz: rstring.

Die Namen sind das Gegenteil dessen, was man vermuten würde; der Klassenkommentar dokumentiert sie als `ristring: str_replace` und `rstring: str_ireplace`, und das Beispiel bestätigt es. Auf einer Entwicklungsinstallation bietet außerdem die Erweiterung owsimpleoperator einen einfachen Operator `str_replace` an (owsimpleoperator).

strtr

PHP `strtr()` in seinen beiden Formen: zwei Zeichenketten (Übersetzung Zeichen für Zeichen) oder ein Hash (ganze Zeichenketten, die längste zuerst). Alles andere protokolliert "Requires (from_string, to_string) or a single associative array".

Parameter	Typ	Erforderlich	Standardwert
from oder ein Hash mit Ersetzungen	string oder hash	ja	keiner
to	string	bei einer Zeichenkette from	keiner

Gibt eine Zeichenkette zurück. Generierte Referenz: strtr.

Template

```
{def $s = 'Fit and healthy. Stay Healthy.'}
ristring( 'healthy', 'strong' ) = {$s|ristring( 'healthy', 'strong' )}
rstring( 'healthy', 'strong' ) = {$s|rstring( 'healthy', 'strong' )}
ristring( array( 'Fit', 'Stay' ), array( 'Be', 'Keep' ) ) = {$s|ristring( array( 'Fit', 'Stay' ),
 array( 'Be', 'Keep' ) )}
strtr( 'aeiou', 'AEIOU' ) = {$s|strtr( 'aeiou', 'AEIOU' )}
strtr( hash( 'Fit', 'Strong', 'healthy', 'happy' ) ) = {$s|strtr( hash( 'Fit', 'Strong', 'healthy
', 'happy' ) )}
```

Ausgabe:

```
ristring( 'healthy', 'strong' ) = Fit and strong. Stay Healthy.
rstring( 'healthy', 'strong' ) = Fit and strong. Stay strong.
ristring( array( 'Fit', 'Stay' ), array( 'Be', 'Keep' ) ) = Be and healthy. Keep Healthy.
strtr( 'aeiou', 'AEIOU' ) = FIt And hEAlthy. StAy HEAlthy.
strtr( hash( 'Fit', 'Strong', 'healthy', 'happy' ) ) = Strong and happy. Stay Healthy.
```

Aufteilen: str_split, str_word_count, str_getcsv und strtok

str_split

PHP `str_split()`: die Zeichenkette in Stücken von length Bytes (Standardwert 1). Gibt ein Array zurück. Generierte Referenz: str_split.

str_word_count

PHP `str_word_count()`: ohne Parameter die Zahl der Wörter; mit format 1 eine Liste der Wörter, mit 2 ein Hash von Position zu Wort. Ein optionaler zweiter Parameter nennt zusätzliche Zeichen, die als Buchstaben gelten.

Parameter	Typ	Erforderlich	Standardwert
format	integer (0, 1 oder 2)	nein	0
characters	string	nein	keiner

Generierte Referenz: str_word_count.

str_getcsv

PHP `str_getcsv()`: eine CSV-Zeile als Array. Das Escape-Zeichen ist standardmäßig `''` (keines), was die in PHP 8.4 als veraltet markierte alte Voreinstellung vermeidet.

Parameter	Typ	Erforderlich	Standardwert
separator	string	nein	,
enclosure	string	nein	"
escape	string	nein	''

Generierte Referenz: str_getcsv.

strtok

PHP `strtok( input, token )`: das erste Token. Weil jeder Operatoraufruf einen neuen Durchlauf von `strtok()` beginnt, kann ein Template nur das erste Stück erhalten; verwenden Sie für alle Stücke `explode`. Parameter: token (string, erforderlich). Generierte Referenz: strtok.

Template

```
'abcdefg'|str_split( 3 ) = {'abcdefg'|str_split( 3 )|implode( ' | ' )}
'one two three'|str_word_count = {'one two three'|str_word_count}
'one two three'|str_word_count( 1 ) = {'one two three'|str_word_count( 1 )|implode( ',' )}
'a,"b,c",d'|str_getcsv = {'a,"b,c",d'|str_getcsv|implode( ' | ' )}
'a;b;c'|str_getcsv( ';' ) = {'a;b;c'|str_getcsv( ';' )|implode( ' | ' )}
'first second third'|strtok( ' ' ) = {'first second third'|strtok( ' ' )}
```

Ausgabe:

```
'abcdefg'|str_split( 3 ) = abc | def | g
'one two three'|str_word_count = 3
'one two three'|str_word_count( 1 ) = one,two,three
'a,"b,c",d'|str_getcsv = a | b,c | d
'a;b;c'|str_getcsv( ';' ) = a | b | c
'first second third'|strtok( ' ' ) = first
```

Ränder und Schreibung: ltrim, rtrim, lcfirst und chunk_split

ltrim und rtrim

PHP `ltrim()` und `rtrim()`: Leerraum oder die angegebenen Zeichen am linken oder am rechten Ende entfernt. Parameter: characters (string, optional). Generierte Referenzen: ltrim, rtrim. Beide Enden auf einmal: trim.

lcfirst

PHP `lcfirst()`: das erste Byte als Kleinbuchstabe. Keine Parameter. Generierte Referenz: lcfirst.

chunk_split

PHP `chunk_split()`: fügt die Endzeichenkette nach jedem Stück ein (standardmäßig 76 Bytes und `"\r\n"`), auch nach dem letzten.

Parameter	Typ	Erforderlich	Standardwert
length	integer	nein	76 wie in PHP
separator	string	nein	"\r\n" wie in PHP

Generierte Referenz: chunk_split.

Template

```
[{'--slug--'|ltrim( '-' )}] [{'--slug--'|rtrim( '-' )}] [{'  both  '|ltrim}]
'Healthy'|lcfirst = {'Healthy'|lcfirst}
'abcdefgh'|chunk_split( 3, '-' ) = {'abcdefgh'|chunk_split( 3, '-' )}
```

Ausgabe:

```
[slug--] [--slug] [both  ]
'Healthy'|lcfirst = healthy
'abcdefgh'|chunk_split( 3, '-' ) = abc-def-gh-
```

Backslashes: addslashes, stripslashes, addcslashes, stripcslashes und quotemeta

Operator	PHP-Funktion	Parameter
addslashes	addslashes(): Backslash vor ', ", \ und NUL	keine
stripslashes	stripslashes()	keine
addcslashes	addcslashes(): Backslash vor den Zeichen einer Liste (Bereiche wie A..Z erlaubt)	characters (erforderlich)
stripcslashes	stripcslashes(): Escape-Folgen im C-Stil wie \t werden zu den Zeichen	keine
quotemeta	quotemeta(): Backslash vor . \ + * ? [^] $ ( )	keine

Keiner davon ist eine Maskierung für HTML- oder JavaScript-Ausgabe; verwenden Sie dafür `wash` (Zeichenketten-Operatoren). Generierte Referenzen: addslashes, stripslashes, addcslashes, stripcslashes, quotemeta.

Im Template-Quelltext wird ein Anführungszeichen innerhalb einer Zeichenkette in einfachen Anführungszeichen als `\'` geschrieben und ein Backslash als `\\`:

Template

```
'It\'s "fit"'|addslashes = {'It\'s "fit"'|addslashes}
'It\\\'s'|stripslashes = {'It\\\'s'|stripslashes}
'FOO[bar]'|addcslashes( 'A..Z' ) = {'FOO[bar]'|addcslashes( 'A..Z' )}
'a\\tb'|stripcslashes|bin2hex = {'a\\tb'|stripcslashes|bin2hex}
'1+1=2?'|quotemeta = {'1+1=2?'|quotemeta}
```

Ausgabe:

```
'It\'s "fit"'|addslashes = It\'s \"fit\"
'It\\\'s'|stripslashes = It's
'FOO[bar]'|addcslashes( 'A..Z' ) = \F\O\O[bar]
'a\\tb'|stripcslashes|bin2hex = 610962
'1+1=2?'|quotemeta = 1\+1=2\?
```

strip_tags und die Registrierungsfalle

strip_tags

PHP `strip_tags()`: entfernt HTML- und PHP-Tags. In `eZTemplateStringsOperator` nennt ein optionaler Parameter die Tags, die erhalten bleiben sollen (eine Zeichenkette wie `'<b><i>'` oder ein Array von Namen).

Parameter	Typ	Erforderlich	Standardwert
erlaubte Tags	string oder array	nein	keiner

Gibt eine Zeichenkette zurück. `strip_tags` ist kein Bereinigungswerkzeug: Es behält die Attribute erlaubter Tags und maskiert `&` nicht. Bereinigen Sie das Ergebnis mit wash, bevor Sie es ausgeben. Generierte Referenz: strip_tags.

Template

```
{def $html = '<p>Fit <b>and</b> <a href="/x">healthy</a></p>'}
strip_tags = {$html|strip_tags}
```

Ausgabe:

```
strip_tags = Fit and healthy
```

Die Registrierungsfalle. Auf einer Entwicklungsinstallation fügt die Erweiterung ezmultiupload `PHPOperatorList[strip_tags]=strip_tags` zu `template.ini` hinzu (`extension/ezmultiupload/settings/template.ini.append.php`). Die Datei `kernel/common/eztemplateautoload.php` des Kernels registriert die Namen aus `PHPOperatorList` über `eZTemplatePHPOperator`, das der PHP-Funktion nur einen Wert übergibt; diese Fassung ignoriert also die erlaubten Tags. Welche der beiden Klassen antwortet, hängt davon ab, was die Anfrage vorher verwendet hat: Die Template-Engine lädt Operatorklassen verzögert, und wird zuerst irgendein Operator von `eZTemplateStringsOperator` verwendet (hier `strlen`), registriert das Laden dieser Klasse alle ihre Namen erneut, einschließlich `strip_tags`, und ab dann antwortet die Fassung mit Parametern. Dieselbe Template-Zeile liefert zwei Ergebnisse:

Template

```
{def $html = '<p>Fit <b>and</b> healthy</p>'}
first use: [{$html|strip_tags( '<b>' )}]

{def $len = 'x'|strlen}
after strlen: [{$html|strip_tags( '<b>' )}]
```

Ausgabe:

```
first use: [Fit and healthy]
after strlen: [Fit <b>and</b> healthy]
```

Wenn Sie die erlaubten Tags brauchen, verlassen Sie sich auf keine der beiden: Entfernen Sie alle Tags und fügen Sie das Markup im Template hinzu, oder registrieren Sie einen eigenen Operator unter einem anderen Namen (Registrierungsreihenfolge und Überschreibungen).

str_shuffle und hebrev

str_shuffle

PHP `str_shuffle()`: die Bytes in zufälliger Reihenfolge (nicht für Geheimes geeignet). Keine Parameter. Generierte Referenz: str_shuffle.

hebrev

PHP `hebrev()`: wandelt logischen hebräischen Text in visuellen Text um, mit einer optionalen Höchstzahl von Zeichen je Zeile (integer). Text ohne hebräische Zeichen wird unverändert zurückgegeben. Generierte Referenz: hebrev.

Template

```
{def $s = 'abcdef'|str_shuffle}
str_shuffle keeps the letters: {$s|str_split( 1 )|unique|count} distinct letters, length {$s|strl
en}
hebrev of a Latin text returns it unchanged: {'plain text'|hebrev}
```

Ausgabe:

```
str_shuffle keeps the letters: 6 distinct letters, length 6
hebrev of a Latin text returns it unchanged: plain text
```

str_increment und str_decrement

Die Klasse nimmt `str_increment` und `str_decrement` (PHP 8.3) in ihre Operatorliste auf, wenn PHP sie hat, aber sie stehen nicht in der Liste `operator_names` von `lib/eztemplate/classes/eztemplateautoload.php`. Wie bei der Falle oben existieren sie also erst, nachdem ein anderer Operator der Klasse sie geladen hat. Für sich allein meldet die Engine sie als nicht registriert:

Template

```
[{'a9'|str_increment}]
```

Ausgabe:

```
[a9]
```

Von der Template-Engine protokolliert:

```
WARNING : Operator 'str_increment' is not registered
```

Nachdem `strlen` die Klasse geladen hat:

Template

```
{def $len = 'x'|strlen}
'a9'|str_increment = {'a9'|str_increment}
'Ab'|str_decrement = {'Ab'|str_decrement}
```

Ausgabe:

```
'a9'|str_increment = b0
'Ab'|str_decrement = Aa
```

Verlassen Sie sich in einem Template, das überall funktionieren muss, nicht auf sie.

Verwandte Seiten

- PHP-Zeichenkettenfunktionen: suchen und vergleichen
- PHP-Zeichenkettenfunktionen: kodieren und formatieren
- PHP-Funktionsoperatoren: `PHPOperatorList`
- Array-Operatoren: die Multibyte-sicheren Operatoren zum Bearbeiten von Zeichenketten
- 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.