Write a template operator in PHP

A task guide for a template operator written in PHP - the eztemplateautoload.php registration, the operator class with operatorList(), namedParameterPerOperator(), namedParameterList() and modify(), input and output...

Source file: `doc/eng-us/help/php-developers/write-a-template-operator-in-php.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/write-a-template-operator-in-php.md

This task guide is for PHP developers who give templates a new operator, such as `{$amount|acmereport_money( 'EUR' )}`. An operator is a PHP class that the template engine (`lib/eztemplate/`) calls with the piped value and the parameters. Template authors find the shipped operators in the template operators section; the application view is Writing template operators.

Contents

1. Step 1: register the autoload path
2. Step 2: list the operator
3. Step 3: the operator class
4. Step 4: use it
5. Input and output
6. Escaping
7. Performance and compiled templates
8. Testing modify()

Step 1: register the autoload path

`extension/acmereport/settings/site.ini.append.php`:

INI

```
<?php /* #?ini charset="utf-8"?
[TemplateSettings]
ExtensionAutoloadPath[]=acmereport
*/ ?>
```

The template engine then reads `extension/acmereport/autoloads/eztemplateautoload.php`, as it does for `ezjscore` (`extension/ezjscore/settings/site.ini.append.php`).

Step 2: list the operator

`extension/acmereport/autoloads/eztemplateautoload.php`:

PHP

```php
<?php
$eZTemplateOperatorArray = array();
$eZTemplateOperatorArray[] = array(
    'script' => 'extension/acmereport/autoloads/acmereporttemplateoperators.php',
    'class' => 'acmeReportTemplateOperators',
    'operator_names' => array( 'acmereport_money' ) );
```

The engine loads the class only when a template uses one of the names, so many operators cost nothing until used. `extension/ezjscore/autoloads/eztemplateautoload.php` registers `json_encode`, `ezscript` and others the same way.

Step 3: the operator class

`extension/acmereport/autoloads/acmereporttemplateoperators.php`:

PHP

```php
<?php
class acmeReportTemplateOperators
{
    function operatorList()
    {
        return array( 'acmereport_money' );
    }

    function namedParameterPerOperator()
    {
        return true;
    }

    function namedParameterList()
    {
        return array(
            'acmereport_money' => array(
                'currency' => array( 'type' => 'string', 'required' => false, 'default' => 'EUR'
),
                'decimals' => array( 'type' => 'integer', 'required' => false, 'default' => 2 ) )
 );
    }

    function modify( $tpl, $operatorName, $operatorParameters, $rootNamespace, $currentNamespace,
                     &$operatorValue, $namedParameters )
    {
        switch ( $operatorName )
        {
            case 'acmereport_money':
            {
                $decimals = max( 0, min( 4, (int)$namedParameters['decimals'] ) );
                $operatorValue = number_format( (float)$operatorValue, $decimals ) . ' ' . $named
Parameters['currency'];
            } break;
        }
    }
}
```

Step 4: use it

```
{$item.amount|acmereport_money()|wash}
{$item.amount|acmereport_money( 'USD', 0 )|wash}
```

Regenerate the autoloads and clear the template caches (`php bin/php/ezcache.php --clear-tag=template`); under Velocity use `exp:velocity deploy`.

Input and output

What	Where it arrives
the piped value (`$x|op`)	`$operatorValue`; set it to the result (it is passed by reference)
named parameters	`$namedParameters`, defaults applied
raw parameters	`$operatorParameters`, unevaluated; use `$tpl->elementValue()` only if you know you need them
the operator's name	`$operatorName`, so one class can serve several operators

An operator used without a piped value (`{acmereport_money( 12 )}`) gets `null` in `$operatorValue`; declare the value as a parameter if that is how it is used.

Escaping

Return values, not HTML, and let the template print them through `|wash`. If an operator must return markup, escape every piece of input in it with `htmlspecialchars( $value, ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401, 'UTF-8' )` (the flags are needed for PHP 8.0, see Escaping output from PHP), and never put inline `style=""` or `<script>` into it (Content Security Policy for your code).

Performance and compiled templates

- Operators run on every render, also inside loops. Keep `modify()` fast and free of database queries; data belongs in a fetch function (Write a fetch function).
- Templates are compiled to PHP. Kernel operators that are pure can describe themselves with `operatorTemplateHints()` so the compiler inlines them (`kernel/common/ezkerneloperator.php` is an example); your operator works without it.
- No side effects: inside a `{cache-block}` the operator does not run again until the block expires.

Testing modify()

`modify()` is a plain method; call it in a DB-less test:

PHP

```php
public function testMoneyFormatsTwoDecimals()
{
    $operators = new acmeReportTemplateOperators();
    $value = 12.5;
    $operators->modify( null, 'acmereport_money', array(), '', '', $value,
                        array( 'currency' => 'EUR', 'decimals' => 2 ) );
    $this->assertSame( '12.50 EUR', $value );
}
```

Related pages

- Template operators (extension developers)
- Template operator pattern
- The template language
- 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.