Getting data into templates: fetch functions and operators

Where a framework passes view models from the controller, Exponential templates can also pull data themselves - fetch functions declared in function_definition.php, template operators for formatting, and variables set...

Source file: `doc/eng-us/help/web-framework-developers/fetch-functions-and-template-data.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/web-framework-developers/fetch-functions-and-template-data.md

This page is for developers who pass everything a template needs from the controller. Exponential supports that (`$tpl->setVariable()` in your view), but much of its own design works the other way: templates **pull** data with `fetch( 'module', 'function', hash( ... ) )`. That is why a content override template can list children, related items or the current user without any PHP. Your extension can offer its own fetch functions and template operators.

Contents

1. Three ways data reaches a template
2. Fetch functions
3. Writing a fetch function
4. Template operators
5. Rules for pulling data

Three ways data reaches a template

Way	Framework analogy	Use for
`$tpl->setVariable( 'items', $items )` in a view	passing variables to `render()`	data of your own module view
`fetch( 'expdemo', 'items', hash( 'limit', 5 ) )` in the template	view composers, Twig functions backed by services, template tags that query	data an override or a shared partial needs wherever it is used
An operator: `$price|expdemo_money`	Twig filters, Blade directives, Django template filters	formatting and small computations

Fetch functions

The kernel's modules offer many: `fetch( 'content', 'node', hash( 'node_id', 2 ) )`, `fetch( 'content', 'list', hash( 'parent_node_id', 2, 'limit', 10 ) )`, `fetch( 'user', 'current_user' )`, `fetch( 'user', 'has_access_to', hash( 'module', 'expdemo', 'function', 'manage' ) )`. Each module declares them in `function_definition.php`, which maps a fetch to a `call_method` and its parameters; `eZFunctionHandler` (`lib/ezutils/classes/ezfunctionhandler.php`) executes them (Fetch functions, Fetch functions: reading data in templates).

Writing a fetch function

From Writing your own fetch function:

PHP

```php
<?php
// extension/expdemo/modules/expdemo/function_definition.php
$FunctionList = array();
$FunctionList['items'] = array(
    'name' => 'items',
    'operation_types' => array( 'read' ),
    'call_method' => array( 'class' => 'expDemoFunctionCollection',
                            'method' => 'fetchItems' ),
    'parameter_type' => 'standard',
    'parameters' => array(
        array( 'name' => 'limit',  'type' => 'integer', 'required' => false, 'default' => 10 ),
        array( 'name' => 'offset', 'type' => 'integer', 'required' => false, 'default' => 0 ) ) )
;
```

The method returns `array( 'result' => $value )`. Objects returned should be `eZPersistentObjects` or offer `attribute()` and `hasAttribute()`, so templates can read `$item.title`.

```
{def $items = fetch( 'expdemo', 'items', hash( 'limit', 5 ) )}
{foreach $items as $item}<li>{$item.title|wash}</li>{/foreach}
{undef $items}
```

Template operators

An operator is a PHP class registered in `extension/<ext>/autoloads/eztemplateautoload.php`, found because the extension is listed in `site.ini [TemplateSettings] ExtensionAutoloadPath[]` (Writing template operators, Functions or operators). Operators must escape what they build or return data the template washes.

Rules for pulling data

- **Permissions are applied by the content fetches** (`content/read` with its limitations). Your own fetches must check permissions themselves, because a template can call them from anywhere.
- **Bound every limit.** A fetch without `limit` on a large tree is the most common performance problem.
- **Mind the caches.** Content views are cached; data from your own tables does not expire them. Use a `{cache-block}` with an `expiry` or clear the cache when your data changes (Caching).
- **Keep the method thin.** Call the same service class your views and commands use, and test it with PHPUnit.

Related pages

- The view layer
- Fetch function pattern
- Template operator pattern
- 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.