Write a fetch function

A task guide for giving templates data through fetch('module', 'function', hash(...)) - the function_definition.php entry, the static method and its result array, parameters, errors, permissions and limits, and how...

Source file: doc/eng-us/help/php-developers/write-a-fetch-function.md Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/write-a-fetch-function.md

This task guide is for PHP developers who want templates to read their data with `fetch( 'acmereport', 'items', hash( 'limit', 5 ) )`. A fetch function is a declaration in the module's `function_definition.php` plus a static PHP method. The template side is described in Fetch functions: reading data in templates.

Contents

1. Step 1: the declaration
2. Step 2: the method
3. Step 3: use it
4. How the kernel calls you
5. The result array
6. Rules

Step 1: the declaration

`extension/acmereport/modules/acmereport/function_definition.php`, beside `module.php`:

PHP

```php
<?php
$FunctionList = array();
$FunctionList['items'] = array(
    'name' => 'items',
    'operation_types' => array( 'read' ),
    'call_method' => array( 'class' => 'acmeReportFunctionCollection',
                            'method' => 'fetchItems' ),
    'parameter_type' => 'standard',
    'parameters' => array(
        array( 'name' => 'status', 'type' => 'string',  'required' => false, 'default' => false )
,
        array( 'name' => 'limit',  'type' => 'integer', 'required' => false, 'default' => 10 ),
        array( 'name' => 'offset', 'type' => 'integer', 'required' => false, 'default' => 0 ) ) )
;
```

The module must be registered (`[ModuleSettings] ExtensionRepositories[]` and `ModuleList[]`); the kernel's own declarations look the same (`kernel/content/function_definition.php`).

Step 2: the method

`extension/acmereport/classes/acmereportfunctioncollection.php`:

PHP

```php
<?php
class acmeReportFunctionCollection
{
    /**
     * @param string|false $status
     * @param int $limit
     * @param int $offset
     * @return array array( 'result' => acmeReportItem[] )
     */
    public static function fetchItems( $status = false, $limit = 10, $offset = 0 )
    {
        $access = eZUser::currentUser()->hasAccessTo( 'acmereport', 'read' );
        if ( $access['accessWord'] === 'no' )
            return array( 'result' => array() );

        $conds = null;
        if ( $status !== false && in_array( $status, array( 'open', 'closed' ), true ) )
            $conds = array( 'status' => $status );

        $items = eZPersistentObject::fetchObjectList( acmeReportItem::definition(), null, $conds,
                                                      array( 'created' => 'desc' ),
                                                      array( 'offset' => max( 0, (int)$offset ),
                                                             'length' => max( 1, min( 100, (int)$
limit ) ) ) );
        return array( 'result' => is_array( $items ) ? $items : array() );
    }
}
```

Then `php bin/php/ezpgenerateautoloads.php -e` and clear the caches.

Step 3: use it

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

How the kernel calls you

`eZModuleFunctionInfo` (`lib/ezutils/classes/ezmodulefunctioninfo.php`) reads the declaration, maps the named parameters of the `hash()` to the method's parameters **in the order of** `parameters`, fills defaults, and calls `call_method`. Keep the method's parameter list in the same order as the declaration.

The result array

You return	The template gets
`array( 'result' => $value )`	`$value`

You return	The template gets
`array( 'error' => array( 'error_type' => 'kernel', 'error_code' => eZError::KERNEL_NOT_FOUND ) )`	`null`
anything without `result` (also `array( 'result' => null ))`	`null`, and an error in the log: the function "did not return a result value"

Return an empty array or `false` inside `result` for "nothing found", so templates can test it without errors in the log.

Rules

- **Check permissions inside the function.** Templates are included from many places; the function is the only place that sees every call.
- **Bound every limit** and cast every parameter, as for any input.
- **Keep it thin.** Call the service your views and commands use.
- **Name parameters like the content fetches** (`limit`, `offset`, `sort_by`), so template authors guess right.
- **Cache-friendly.** Fetches inside `{cache-block}` run only when the block expires; do not depend on side effects.
- **Test the static method** like any other (PHPUnit without a database).

Related pages

- Writing your own fetch function (application creators)
- Fetch functions reference
- Fetch function 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.