Write your first module view

A step-by-step task guide - create an extension with a module, declare a view and a policy function in module.php, implement the view as a class, render a template, give a role the policy, regenerate autoloads and open...

Source file: doc/eng-us/help/php-developers/write-your-first-module-view.md Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/write-your-first-module-view.md

This task guide is for PHP developers who want a new page at an address of their own. At the end you have an extension `acmereport` with a module `acmereport` and a view `list` at `/acmereport/list`, written as a class, protected by a policy, rendering a template. Replace `acme` with your own prefix (Class naming and the exp prefix).

Contents

1. Before you start
2. Step 1: the extension and its activation
3. Step 2: register the module
4. Step 3: declare the view and the policy
5. Step 4: the view class
6. Step 5: the view file
7. Step 6: the template
8. Step 7: autoloads, caches, a policy
9. Step 8: check it
10. What to do next

Before you start

You need a development checkout with an installation (Setting up a PHP development checkout) and an administrator account to assign a policy. The files you create:

```
extension/acmereport/
  settings/site.ini.append.php
  settings/module.ini.append.php
  modules/acmereport/module.php
  modules/acmereport/list.php
  classes/runnable/views/acmereport/list.php
  design/standard/templates/acmereport/list.tpl
```

Step 1: the extension and its activation

Create the directory `extension/acmereport/` and activate it for the installation in `settings/override/site.ini.append.php`:

INI

```ini
[ExtensionSettings]
ActiveExtensions[]=acmereport
```

Step 2: register the module

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

INI

```ini
<?php /* #?ini charset="utf-8"?
[ModuleSettings]
ExtensionRepositories[]=acmereport
ModuleList[]=acmereport
*/ ?>
```

`ExtensionRepositories[]` tells the kernel to look for modules in `extension/acmereport/modules/`.

Step 3: declare the view and the policy

`extension/acmereport/modules/acmereport/module.php`:

PHP

```php
<?php
$Module = array( 'name' => 'Acme report' );

$ViewList = array();
$ViewList['list'] = array(
    'script' => 'list.php',
    'functions' => array( 'read' ),
    'params' => array(),
    'unordered_params' => array( 'offset' => 'Offset' ) );

$FunctionList = array();
$FunctionList['read'] = array();
```

- `functions` names the policy a user needs: `acmereport/read`.
- `unordered_params` maps `/acmereport/list/(offset)/20` to `$Params['Offset']`.
- Every key is described in module.php reference.

Step 4: the view class

`extension/acmereport/classes/runnable/views/acmereport/list.php`:

PHP

```php
<?php
namespace Exponential\View\Extension\Acmereport\Acmereport;

class ListView extends \Exponential\Runnable\ModuleView
{
    public function run( array $scope )
    {
        $params = $scope['Params'];
        $offset = isset( $params['Offset'] ) ? max( 0, (int)$params['Offset'] ) : 0;
```

```
        $items = \eZPersistentObject::fetchObjectList( \acmeReportItem::definition(), null, null,
                                                       array( 'created' => 'desc' ),
                                                       array( 'offset' => $offset, 'length' => 20
 ) );

        $tpl = \eZTemplate::factory();
        $tpl->setVariable( 'items', is_array( $items ) ? $items : array() );
        $tpl->setVariable( 'offset', $offset );

        return array(
            'content' => $tpl->fetch( 'design:acmereport/list.tpl' ),
            'path' => array( array( 'text' => \ezpI18n::tr( 'extension/acmereport', 'Reports' ),
'url' => false ) ) );
    }
}
```

`list` is a reserved word in PHP, so the class is named `ListView`, as the kernel names its own (`Exponential\View\Kernel\Section\ListView`); the namespace follows `Exponential\View\Extension\<Ext>\<Module>\<View>` (Module views as classes). The `acmeReportItem` class is the persistent object of Persistent object definitions in depth; for a first try, pass an array instead.

Step 5: the view file

`extension/acmereport/modules/acmereport/list.php`, the file `module.php` names, is one line:

PHP

```
<?php
return \Exponential\View\Extension\Acmereport\Acmereport\ListView::main( __FILE__, get_defined_va
rs() );
```

`get_defined_vars()` hands the class what `eZProcess::runFile()` defined: `$Params` (with the module as `$Params['Module']`) and the rest.

Step 6: the template

`extension/acmereport/design/standard/templates/acmereport/list.tpl`:

```
<h1>{'Reports'|i18n( 'extension/acmereport' )}</h1>
{if $items|count}
<ul>
{foreach $items as $item}
    <li>{$item.status|wash}: {$item.amount|wash}</li>
{/foreach}
</ul>
{else}
<p>{'No reports yet.'|i18n( 'extension/acmereport' )}</p>
{/if}
```

Register the design so the template is found, in `extension/acmereport/settings/design.ini.append.php`:

INI

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

Every value goes through `|wash` (Escaping output from PHP).

Step 7: autoloads, caches, a policy

Shell

```
php -l extension/acmereport/classes/runnable/views/acmereport/list.php
php bin/php/ezpgenerateautoloads.php -e
php bin/php/ezcache.php --clear-all
```

In the administration, **Roles and policies**: give a test role the policy `acmereport/read` (module `acmereport`, function `read`) and assign the role to your test user. Under Velocity, run `php bin/php/console exp:velocity deploy` instead of the cache command, so the workers load the new class.

Step 8: check it

Open `/acmereport/list` on the site's siteaccess, signed in as the test user.

You see	Cause
the module or view was not found	`ModuleList[]` or `ExtensionRepositories[]` missing, or the INI cache not cleared
access denied	the user lacks `acmereport/read`
`Class ... not found`	autoloads not regenerated, or the namespace and class name differ from the view file
the template was not found	`DesignExtensions[]` missing, or the path under `templates/` differs
an empty page	a PHP error: enable the debug output and read `var/log/error.log`

What to do next

- Add a form with actions and the form token: Custom modules and views.
- Give templates the data with a fetch function: Write a fetch function.
- Test the logic without a database: PHPUnit without a database.
- Do it faster next time with **Setup > RAD** (RAD tools).

Related pages

- Modules and views (extension developers)
- View results, templates and the pagelayout
- Policy functions and limitations
- 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.