Exponential for PHP developers

What a PHP developer meets in Exponential - a large, mature PHP code base with its own conventions, generated class maps, layered INI settings, persistent objects and a template engine - and how it differs from a...

Source file: `doc/eng-us/help/php-developers/exponential-for-php-developers.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/exponential-for-php-developers.md

This page is for PHP developers who are new to the Exponential code base. It explains what kind of PHP application Exponential is, which conventions you will meet in the first hour, and where they differ from the frameworks most PHP developers know today. The detailed pages of this section build on it.

Contents

1. What the code base is
2. Five things that surprise PHP developers
3. The building blocks you will write
4. What the project checks on every change
5. Where to go next

What the code base is

Exponential is an open source content management system and PHP application framework, licensed under the GNU GPL, version 2 or later (`composer.json`, `"license": "GPL-2.0-or-later"`). The code base has been developed since 1999, first under the name eZ Publish, a product its creator discontinued after almost 18 years; 7x kept it alive, and the product has been called Exponential since August 2025 (Who is 7x?).

For a PHP developer that history matters in one way: the code is older than Composer, PSR-4 and namespaces, and it kept its own conventions. They are consistent and documented, and the project's checks enforce them on new code.

Part	Directory	What it holds
Kernel	`kernel/`	content model, modules and views, datatypes, workflows, users and roles, the request kernel (`kernel/private/classes/ezpkernelweb.php`)
Libraries	`lib/`	general building blocks: `lib/ezutils` (`eZINI`, `eZDebug`, `eZSys`, `eZHTTPTool`, `eZExecution`), `lib/ezdb` (`eZDB` and the drivers), `lib/eztemplate` (the template engine), `lib/ezfile`, `lib/ezi18n` and more
Extensions	`extension/`	everything added on top, shipped or your own: modules, datatypes, operators, designs, settings
Settings	`settings/`	the INI defaults; installations override them in `settings/override/` and `settings/siteaccess/`

Part	Directory	What it holds
Entry points	`index.php`, `bin/php/*.php`, `runcronjobs.php`	the web front controller, the command line scripts and the cronjob runner

A tour of every directory is on A tour of the source tree.

Five things that surprise PHP developers

1. **Class maps, not PSR-4.** Classes are found through generated arrays: `autoload/ezp_kernel.php` for the kernel and `var/autoload/ezp_extension.php` for the extensions, written by `php bin/php/ezpgenerateautoloads.php`. A new class that is not in the map does not exist (Autoloads and class maps).
2. **The kernel style has no type declarations.** `phpcs.xml.dist` refuses `declare( strict_types=1 )`, scalar parameter types, return types and property types in new code, to match the existing code and keep PHP 8.0 working (Why the code has no type declarations).
3. `array( ... )`**, not** `[ ... ]`**, and spaces inside parentheses.** `foo( $a, $b )`, `if ( $a )`, the opening brace of a function on its own line (The coding standard).
4. **Settings are INI files in layers.** `eZINI::instance( 'site.ini' )->variable( 'Block', 'Key' )` reads the value after the kernel defaults, the extensions, the siteaccess and the installation's overrides were merged (Reading settings with eZINI).
5. **Much state is static.** Singletons such as `eZINI::instance()`, `eZDB::instance()` and in-memory caches live in static properties. Under PHP-FPM each request starts fresh; under Velocity's persistent workers they survive between requests unless the kernel resets them (Static state traps).

The building blocks you will write

You want	You write	Page
a new screen at `/<module>/<view>`	a module with views	Write your first module view
a command line tool	a class extending `Exponential\Runnable\Command`	Write a console command
scheduled work	a cronjob part	Write a cronjob part
data for templates	a fetch function	Write a fetch function
a template filter or helper	a template operator class	Write a template operator in PHP
a new field type	a class extending `eZDataType`	Write a datatype
a step in a workflow	a class extending `eZWorkflowEventType`	Write a workflow event type
data in your own table	a class extending `eZPersistentObject`	Persistent object definitions in depth
JSON for your own JavaScript	an ezjscore server function	ezjscore JSON endpoints in PHP

You want	You write	Page
a reaction to something the kernel does	an event listener	Events, listeners and filters

What the project checks on every change

Check	Tool	Page
PHP syntax, also with PHP 8.0	`php -l`, `make lint80`	Writing code that runs on PHP 8.0
Code style of changed files	`make phpcs` (`phpcs.xml.dist`)	The coding standard
Static analysis	`make phpstan` (level 1 with a baseline)	Static analysis with PHPStan
Unit tests	`vendor/bin/phpunit`	PHPUnit without a database
The file manifest	`make manifest-check`	Git hooks and make targets

Where to go next

- Follow The PHP developer's reading path in order.
- Set up a checkout: Setting up a PHP development checkout.
- Learn the request: How a request runs in PHP.

Related pages

- PHP Developers: start here
- Kernel architecture
- Application creators: start here, the extension developer book
- Software Engineers: start here, Web Framework Developers: start here
- 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.