Extensions you will write

For CMS developers, the extensions a typical site project needs, seen from the project. These are a site extension with design and settings, and optionally a code extension with operators, fetch functions, modules...

Source file: `doc/eng-us/help/cms-developers/extensions-you-will-write.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/cms-developers/extensions-you-will-write.md

This page is for developers planning the code of a site project. In Exponential, everything a project adds lives in **extensions** under `extension/`: designs, settings, PHP code, templates, translations. The kernel and the shipped extensions stay untouched, which is what makes updates safe. A typical project needs one or two extensions of its own.

Contents

1. The usual set
2. The site extension
3. A code extension
4. What a code extension can add
5. Activating and loading
6. Version control
7. Fast starts: the RAD tools
8. Where the book goes deeper

The usual set

Extension	Holds	Example name
Site extension	the design, override rules, siteaccess settings, image aliases, custom tags, classes as packages	`exp_mysite`
Code extension (when needed)	PHP: operators, fetch functions, modules, workflow events, cronjobs, console commands	`exp_mysite_tools`

Small projects keep both in one extension. Split them when the code is reused by other sites, or when designers and PHP developers work on different release cycles. Name them with your project's prefix and `exp` naming in new code (Naming prefixes).

The site extension

```
extension/exp_mysite/
  extension.xml, ezinfo.php, composer.json, README.md
  settings/
    design.ini.append.php            DesignExtensions[]=exp_mysite
    site.ini.append.php              extension-wide settings
    image.ini.append.php, content.ini.append.php
    siteaccess/<public sa>/          site.ini, override.ini ... of the public siteaccess
  design/exp_mysite/
    templates/  override/templates/  stylesheets/  javascript/  images/
  translations/<locale>/translation.ts
  packages/                          classes exported as packages
```

Design work: Your own design extension, Design extension anatomy and the designers' Theme extensions: start here.

A code extension

```
extension/exp_mysite_tools/
  extension.xml, ezinfo.php, composer.json
  classes/                    PHP classes, autoloaded
  autoloads/                  eztemplateautoload.php for template operators
  modules/exp_mysite_tools/   module.php, views, function_definition.php
  eventtypes/event/<type>/    workflow event types
  cronjobs/                   cronjob scripts
  bin/php/                    command line scripts
  settings/                   the registrations and your own .ini
  tests/
```

Every folder and the setting that makes the kernel find it: Extension directory layout.

What a code extension can add

You need	Add	Read
A computed value or a formatter in templates	a template operator	Template operators
Data no fetch returns	a fetch function	Fetch functions
A page with its own address, a form handler, a JSON endpoint	a module view with a policy	Modules and views
A step in publishing or checkout	a workflow event type	Workflows and triggers
Scheduled work	a cronjob	Cronjobs
A field with its own storage and editing	a datatype	Choosing datatypes
Data that is not content	a table with `eZPersistentObject`	Persistent objects and tables
An import or maintenance job	a console command or script	CLI scripts and console commands

Try settings, templates and existing extensions before code; the book's chapter When settings are not enough helps decide.

Activating and loading

1. Add the extension to `site.ini` `[ExtensionSettings]` `ActiveExtensions[]` (or `ActiveAccessExtensions[]` in one siteaccess):

   Shell

   ```
   ./console exp:ini add 'site.ini/ExtensionSettings/ActiveExtensions[]' exp_mysite global --allow-root-user
   ```

2. Regenerate the autoloads after adding or renaming classes: `php bin/php/ezpgenerateautoloads.php -e`.
3. Clear the INI cache and restart the PHP workers that serve the site (Deploying a site).

Order matters: an extension earlier in `ActiveExtensions[]` wins in settings and in the design chain (Activating an extension, Extension load order).

Version control

Each extension of yours is its own Git repository, or part of your project repository; the kernel and the shipped extensions come from Composer. Commit templates, settings, packages and code; never commit `settings/override/`, `var/` or credentials (Develop with Git, Development environment). Give the extension an `extension.xml` and `ezinfo.php` with a version, so the administration's about page and the upgrade checks can tell which release is installed (extension.xml, ezinfo.php).

Fast starts: the RAD tools

The administration has wizards that write a working skeleton: a design extension, a module, a template operator, a datatype, a workflow event, a settings extension, a content extension with classes (RAD tools, The RAD tools tour). Read what they generate; it is your code from then on.

Where the book goes deeper

The book Extension development covers all of this in depth, from the first extension to releasing it. Use its Reading paths: the builder path (settings, templates, overrides) and the PHP developer path fit a CMS developer best. Releasing an extension to others follows strict rules on versions and tags (Releasing).

Related pages

- Extension points at a glance
- What an extension is
- Module 7: Extensions
- 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.