Caching for CMS developers

For CMS developers, every cache that can stand between a change and the page. It covers the view cache and what it is keyed by, cache blocks in the pagelayout, which publish clears what, the template, override and INI…

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

This page is for developers who want fast pages without stale ones. Exponential caches at several levels; most work without you, and the ones that do not need a decision in your templates. The two questions you answer on every project are: *what may be shared between visitors*, and *which cache must go after which change*.

Contents

1. The layers
2. The view cache
3. Cache blocks
4. What publishing clears
5. Caches of code and settings
6. Whole pages: HTTP cache and Velocity
7. Which cache after which change
8. Visitor-specific output
9. Seeing what is cached

The layers

Layer	Holds	Shipped
View cache	the rendered view of a node (`full` and other cached modes)	on: `site.ini [ContentSettings] ViewCaching=enabled`
Cache blocks	parts of templates you wrap in `{cache-block}`	where your templates use them
Template compile cache	templates compiled to PHP	on
Override, INI, translation caches	resolved settings and template lookups	on
Image aliases	scaled images	made on first use
SQL query cache	results of repeated queries	on
HTTP cache	whole pages, role-aware, for any engine	see its status
Velocity response cache	whole answers on the Velocity server	`[CacheSettings]` of `velocity.ini`

Layer	Holds	Shipped
Static cache	pages written as files for the web server	off

Every layer, with where it lives and how it is cleared: Caching layers.

The view cache

`CachedViewModes=full;sitemap;pdf`: these view modes, opened as pages, are stored. The pagelayout is **not** in the view cache; it runs on every request around the cached content. A cached view is keyed by node, view mode, language, the user's roles and limitations, view parameters and more (`kernel/classes/eznodeviewfunctions.php`), so visitors with different rights get different copies.

A full view template can set its own lifetime: `{set-block scope=root variable=cache_ttl}400{/set-block}` keeps that page for 400 seconds, useful for a page that compares with `currentdate()`. `ViewCacheTweaks[<node id>]` in `site.ini` has per-node options such as caching per user (Caching for designers).

Cache blocks

Wrap expensive parts of the pagelayout (menus, footers) in a cache block, with every value that changes the output in its keys:

Template

```
{cache-block keys=array( $module_result.uri, $current_user.role_id_list|implode( ',' ) ) subtree_
expiry=$menu_root_node_id}
    {include uri='design:parts/main_menu.tpl'}
{/cache-block}
```

Without `keys` there is one copy for every page and visitor. By default a block expires after 7200 seconds and on every publish; `subtree_expiry` expires it only for publishes below a node, `ignore_content_expiry` never on publish (Cache block).

What publishing clears

Publishing clears the view cache of the object's nodes, their parents, and of related objects by the relation types in `viewcache.ini [ViewCacheSettings] ClearRelationTypes[]` (`common`, `reverse_common`, `reverse_embedded` shipped). `SmartCacheClear=disabled` is shipped; enabled, rules per class clear more, for example the front page when an article is published. A front page that lists the newest articles is cleared when one of them is published only if a rule or a relation connects them, so give list pages a short `cache_ttl` or a smart clear rule.

Caches of code and settings

You changed	Clear
an INI file	`php bin/php/ezcache.php --clear-tag=ini --allow-root-user`
an existing template	`--clear-id=template` (nothing while `TemplateCompile=disabled` on a development machine)

You changed	Clear
a new template or override rule	`--clear-tag=ini` and `--clear-id=template-override`, then `content`
a `.ts` translation	`--clear-tag=i18n`
an image alias	`--clear-tag=ini` and `--clear-id=imagealias`
a stylesheet or script in the packer	the packer and template-block caches (The packer)
a PHP class	regenerate autoloads and restart the PHP workers (Deploying a site)

Velocity keeps settings and classes in its workers: after INI or PHP changes, restart it (Caches: what each one holds).

Whole pages: HTTP cache and Velocity

The HTTP cache and Velocity's response cache keep whole answers; a page served from them never reaches PHP. Clear Velocity's after template changes with `./console exp:velocity cache clear --apply`, and always **after** the PHP workers run new code, or a page rendered by old code is cached again (HTTP cache, Velocity response cache). Static cache is off in new installations; leave it off unless you have measured a need.

Which cache after which change

The administration's **Setup** > **Caches** lists every cache with its id, size and the command that clears it, and `./console exp:cache clear --id=<id> --dry-run` shows what would go. When unsure, read Why doesn't my change appear? and Caches after a change before clearing everything: `--clear-all` on a busy site makes every visitor wait while caches are rebuilt.

Visitor-specific output

The most common caching bug is a visitor's name, basket or "edit" button cached and shown to the next visitor.

- Put visitor-specific parts in the pagelayout (not view-cached), keyed in a cache block by user, or leave them uncached.
- In a full view, print only what is the same for everyone with the same roles.
- Never key a cache block by something a visitor controls without limits (a free text parameter): it fills the disk.

Seeing what is cached

Shell

```
./console exp:cache status --allow-root-user
./console exp:cache clear --id=content --dry-run --allow-root-user
```

`status` prints one line per layer; the debug output shows whether a view came from the cache. Performance work starts here: Performance.

Related pages

- Template compilation and cache
- Cache tags and clearing
- Caching and templates for builders
- 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.