Caching: the layers and how to invalidate them

The caches of Exponential for developers used to a cache component with pools and tags - compiled templates, INI and override caches, the content view cache per role, template cache blocks, data caches through the…

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

This page is for developers who use a cache component (Symfony Cache, Laravel's `Cache`, Django's cache framework, Rails' `Rails.cache`). Exponential has several caches, each with an id and tags, handled by `eZCache` (`kernel/classes/ezcache.php`), and most of them are invalidated by the kernel when content is published. The usual reason "my change does not show" is one of these layers (Caches).

Contents

1. The layers
2. Clearing
3. Fragment caching in templates
4. Caching data in PHP
5. Under persistent workers
6. Compared with other frameworks

The layers

Layer	Keeps	Cleared by
Compiled templates (`template`)	`.tpl` compiled to PHP	template changes, `--clear-tag=template`
Template override cache (`template-override`)	which file an override rule resolves to	a new template file needs it cleared
INI caches (`ini`, `global_ini`)	parsed settings	`--clear-tag=ini`
Content view cache (`content`)	rendered content views, per permission context	publishing the node and related content (`eZContentCacheManager`)
Template block cache (`template-block`)	`{cache-block}` fragments	publishing (by default), expiry
Role-aware HTTP cache (`exphttpcache`)	whole pages per permission context; off until `httpcache.ini` `Enabled=enabled`	content changes (The role-aware HTTP cache)
Velocity response cache	public pages inside the server process; off until enabled	its own rules, `./console exp:velocity cache clear --apply` (Velocity response cache)

Layer	Keeps	Cleared by
Others	`urlalias`, `imagealias`, `translation`, `design_base`, `user_info_cache`, `querycache` ...	their ids and tags

Content view caching is `site.ini [ContentSettings] ViewCaching` (`enabled` shipped). The full list with ids and tags: Caches guide.

Clearing

Shell

```
php bin/php/ezcache.php --clear-tag=ini --allow-root-user
php bin/php/ezcache.php --clear-id=template-override --allow-root-user
./console exp:cache status
./console exp:velocity deploy          # the usual way after a code change: everything in the rig
ht order
```

`exp:velocity deploy` clears caches in an order chosen so that a page rendered by old code is never cached again (Servers and deployment). An extension can add its own cache to the list with `site.ini [Cache] CacheItems[]` and a `[Cache_<id>]` block, so `ezcache.php` and **Setup > Caches** can clear it.

Fragment caching in templates

```
{cache-block keys=array( 'expdemo-stats', $current_user.contentobject_id ) expiry=300}
  {def $stats = fetch( 'expdemo', 'stats', hash() )} ... {undef $stats}
{/cache-block}
```

Data from your own tables does not trigger the publish clear: choose an `expiry`, or use `ignore_content_expiry` and clear the `template-block` cache yourself when your data changes. Key by everything the output depends on, including the user when the fragment is personal (Template caches).

Caching data in PHP

There is no PSR-6 or PSR-16 cache pool in the kernel. Expensive results are stored through the cluster file handler, so they work on one server and on a cluster:

PHP

```
$file = eZClusterFileHandler::instance( eZSys::cacheDirectory() . '/expdemo/stats.cache' );
$stats = $file->processCache(
    function ( $path, $mtime ) { return json_decode( (string)file_get_contents( $path ), true );
},
    function ( $path ) {
        $stats = expDemoStats::compute();
        return array( 'content' => $stats, 'binarydata' => json_encode( $stats ), 'scope' => 'exp
demo-stats' );
    },
    300 );
```

While one process generates, others wait instead of generating the same data (Caching in applications). Cache files must be written through `eZClusterFileHandler::instance()`, not

with direct file calls. With the `sevenx_valkey_cache` extension an installation keeps the content view cache and the compiled INI cache in Valkey or Redis (sevenx_valkey_cache).

Under persistent workers

Velocity keeps PHP processes between requests: a static variable survives into the next request. Never keep user-specific data in statics; process memory is not shared between workers, so it is not a shared cache either.

Compared with other frameworks

Framework	Concept	Exponential
Symfony	cache pools, tags, HTTP cache with ESI	cache ids and tags in eZCache, the view cache, the HTTP caches
Laravel	`Cache::remember()`	`processCache()` with a generate callback
Django	per-view cache, template fragment cache	content view cache, `{cache-block}`
Rails	fragment and Russian-doll caching	`{cache-block}` with keys

The difference that matters: content caches are invalidated by publishing, automatically and per node; your own data needs explicit expiry or clearing.

Related pages

- HTTP caching and reverse proxies
- View cache and cache blocks
- Cache console
- 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.