Examples

Working examples for Exponential 6. Web server configurations that were checked, a cookbook of complete tasks, and more than two thousand reference pages generated from the code, covering template operators, fetch…

Source file: `doc/eng-us/examples/README.md` Last changed 2026-11-06
https://doc.exponential.earth/learninminutes/exphelp/viewer/examples/README.md

This section has examples you can copy into a real installation. There are three kinds:

> **Warning**
>
> `.htaccess` **slows down every request.** When `.htaccess` files are enabled (`AllowOverride` anything but `None`), let alone used, Apache looks for them and reads and evaluates them on every request, in every directory of the path, and page loading times go up drastically (measured). The only real fix is to switch the feature off for the virtual host: `AllowOverride None` in the HTTPS virtual host, and the rules in the virtual host's own configuration as mod_rewrite rules. That is far faster in production speed tests and well worth the time it takes to optimize a production configuration. Switch over right away: copy the rules of the example HTTPS virtual host into your virtual host (with the static cache, also `staticcache_offload.conf`), set `AllowOverride None`, run `apachectl configtest`, reload Apache and test the site. The same rules for other servers: nginx, LiteSpeed, Caddy, IIS, Velocity (built in); where virtual host rules go in a control panel: Plesk, cPanel, DirectAdmin, ISPConfig. **On LiteSpeed or OpenLiteSpeed,** the server's own page cache can also answer anonymous visitors and crawlers without PHP: Page caching with LiteSpeed and OpenLiteSpeed.

- **Web server examples**: complete configurations for Exponential Velocity, Apache (`.htaccess` and virtual host), nginx, Caddy, FrankenPHP, LiteSpeed, IIS and PHP-FPM. Each one routes requests the same way as the shipped `.htaccess_root`. Each was syntax-checked, and most were also tested on a scratch server.
- **Cookbook**: complete tasks from start to finish. A content class, a template override, your own template operator, datatype, module, workflow event and cronjob, an extension skeleton, JSON calls, several sites in one installation, translations, caching and deploying with Velocity.
- **Reference examples, generated from the code**: one page per item, each with an example derived from the definition the code ships. The list is below.

The rules for writing code (naming, file layout, security, templates without inline styles or scripts) are in Standards and patterns. This section shows them in use and does not repeat them.

Web servers and hosting

Page	What it shows
Web server examples	The overview: which server, which file, and how it was checked
Apache .htaccess	`htaccess`: the shipped rules, explained one by one

Page	What it shows
Apache virtual host	`ezpublish.conf`: HTTPS, HTTP/2, PHP-FPM and the rules in server context
nginx	A server block with the same routing, HTTPS and headers
Caddy	Caddy with PHP-FPM, routed like the generated FrankenPHP configuration
FrankenPHP	Run by Velocity, or on its own
LiteSpeed and OpenLiteSpeed	`.htaccess` under LiteSpeed, and a virtual host for OpenLiteSpeed
IIS	`web.config` with URL Rewrite
PHP-FPM	A pool for one site
Velocity	`velocity.ini` overrides, the `/etc/vc` tree and a systemd unit
Demo mode	`EXP_DEMO_MODE=1` for a public demo: `.htaccess`, virtual host, PHP-FPM pool, nginx, a systemd drop-in, a crontab, a container environment and a rebuild script
URL map examples	URL maps of an old site in CSV, JSON, Apache and nginx format, with query-string redirects, temporary redirects and wildcards, for `exp:url:import`

`staticcache_offload.conf` adds rules that serve pages from the static cache directly. See Apache .htaccess.

The shipped `.htaccess_root` (or `.htaccess_root_static`) and `config.php-RECOMMENDED` are put into place with `./console exp:setup:htaccess --apply` and `./console exp:setup:config --apply`, and `config.php` is edited with `./console exp:config:set`: see Setting up .htaccess and config.php from the console and the generated pages exp:setupfiles and exp:configphp.

Reference examples generated from the code

Area	What each page has
Template operators	The parameters (type, required, default), an example call, and the first use in a shipped template
Template functions	The class, inner functions, how it finds its template, and the first use in a shipped template
Fetch functions	`fetch( 'module', 'function', hash( ... ) )` with every parameter, the same call in PHP, and a real use
Module views	The address with ordered and unordered parameters, the policy functions it needs, and a link to it
Policy functions and limitations	Each module's policy functions, their limitations, the views that check them, and examples
INI settings	One page per block: each setting with the comment of the shipped file, its default, and an override
Datatypes	The identifier, what it stores, templates that use it, and the shipped view and edit templates
Workflow event types and triggers	Each event type, the triggers it accepts, and every trigger point
Cronjob scripts and parts	What each script does, its parts, and how to run and schedule it

Area	What each page has
Console commands	Every `./console exp:*` command with its help text, options and aliases
Extension points	INI lists of handlers, classes and directories an extension can add to
Migration step types	Every step type of YAML migrations with its modes, keys, reference values and an example

The help lists these pages in folders. A folder opens in the navigation once you are on one of its pages; its overview page links to every page in it. They rank below the written guides in the search, so a question finds the explanation first and the reference second.

What "derived from the code" means

- **Read from the definitions.** Names, parameters, types, defaults, policy functions, limitations, INI values and their comments, and option lists are read from the files the kernel loads, such as `eztemplateautoload.php`, `function_definition.php`, `module.php`, `operation_definition.php`, the shipped INI files, the datatype and workflow event classes, `cronjob.ini` and the command classes. Each page names its sources at the bottom.
- **Placeholders.** Values in an example call, such as `1` for an integer or `'text'` for a string, only show the form. Each page says so.
- **Heuristics.** Some facts are guesses and are marked as such: the storage columns a datatype uses (found by reading its class) and the extension points (found by the setting's name).
- **First use in a shipped template.** This is the first match in the templates of the repository, in path order. It is real code, but not always the simplest example.
- **Only the shipped code.** Only files tracked in the repository are read: the kernel, its libraries and the bundled extensions. Other installed extensions, local overrides and the database are left out. This holds for settings a source asks for while it is read (a `module.php` that lists the values of an INI setting, the PHP operators of `template.ini`): they come from `settings/` and the bundled extensions' settings, never from an override, a siteaccess, the INI cache or an extension outside the repository, so an installation, a git worktree and CI get the same pages. The object state limitations are the one place where the kernel would read the database. Their pages say where those limitations come from instead.

How the generated pages are made

They come from one command, run from the installation root:

Shell

```
./console exp:help:examples                # write every area; only changed files are written
./console exp:help:examples --list         # the areas and how many pages each has
./console exp:help:examples --check        # change nothing; exit 1 if a page is missing, differ
ent or stale
./console exp:help:examples --area=datatypes,cronjobs
./console exp:help index --rebuild         # then rebuild the help index
```

The same command writes the German pages with `--language=ger-de`, into `doc/ger-de/beispiele/`, and lists them in `doc/ger-de/.paths.json`:

Shell

```
./console exp:help:examples --language=ger-de
./console exp:help:examples --language=ger-de --check
./console exp:help:examples --language=ger-de --worklist=var/tmp/todo.json
```

The output is deterministic. The same sources always produce the same files: input is sorted, and pages contain no dates and no host names. Run the command after a change to an operator, a fetch function, a view, a policy, an INI file, a datatype, a workflow event, a cronjob or a command, then commit the changed pages with that change. `--check` in a review or CI job catches pages that were not regenerated.

A translation reads its text from `doc/<language>/.examples/`: `strings.json` holds the fixed text of the pages (headings, labels, sentences) with their `{placeholders}`, and `descriptions/*.json` the comments and descriptions quoted from the sources, by item (such as `ini:settings/site.ini [Block] Setting`), each with the English text it translates. A translation is used only while its English text is the source's; the worklist of `--worklist` lists the texts to translate again. Code, INI keys and values, commands and identifiers are never translated. The English pages do not depend on any translation.

Every generated page has `generated: exp:help:examples` in its front matter. The command only replaces or removes pages with that marker, inside the areas listed above. It never touches the hand-written pages: this one, the web server examples and the cookbook. To change a generated page, change its source and run the command again. Edits to the page itself are lost the next time the command runs.

The code is `kernel/classes/help/exphelpexamplesgenerator.php` (class `expHelpExamplesGenerator`), the translation `kernel/classes/help/exphelpexamplestranslation.php` (class `expHelpExamplesTranslation`), and the command is `kernel/private/classes/commands/exphelpexamples.php`. To add an area, give the class an `area<Name>()` method and an entry in `AREAS`.

Help settings for this section

`settings/exphelp.ini`:

- The category `examples`: `[CategorySettings] Categories[]`, `CategoryNames[examples]`, `CategoryIcons[examples]`.
- The page type `generated` for the generated pages: `[TypeSettings] TypeNames[generated]`. It ranks lower in the search through `[SearchSettings] TypeBoost[generated]`, and stays off the dashboard's audience cards through `[DashboardSettings] AudienceExcludeTypes[]`.
- The folded folders of the navigation: `[NavigationSettings] FoldedFolders[]`.

Related pages

- Standards and patterns: the rules the examples follow.
- Serving the site and Security hardening.
- Templates and design, Roles and policies, Workflows, Cronjobs.
- The help system.

- Free documentation: this page is free to copy, translate and improve.

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.