Search

For CMS developers, how search works on an Exponential site. It covers the built-in engine and Solr through eZ Find, what gets indexed, delayed indexing and cron, the search page and its template, searching from…

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

This page is for developers who build the search of a site. Exponential ships a search engine that keeps its index in the site's database and needs nothing else; Solr through the `ezfind` extension is the option for sites that need relevance tuning, facets or large volumes. The content model decides what is searchable in both.

Contents

1. Two engines
2. What is indexed
3. When content is indexed
4. The search page
5. Searching from templates
6. Searching from PHP
7. Rebuilding the index
8. Choosing for a project

Two engines

	Built-in	Solr through eZ Find
Setting	`site.ini [SearchSettings] SearchEngine=eZSearchEngine` (shipped)	the `ezfind` extension active, with a Solr server
Index lives in	the database	the Solr server
Extra service	none	Solr, local or remote
In a new installation	yes	no; `composer.json` suggests `se7enxweb/ezfind`

`settings/solr.ini [SolrSettings] Enabled=false` is shipped; `./console exp:solr status` says whether a Solr is configured and answers (Search backends). This repository does not state which Solr versions eZ Find needs; take that from the extension's own repository.

What is indexed

Every attribute whose class attribute is **searchable** (`is_searchable`) is indexed through its datatype; untick it for fields that should never match (internal notes, codes). The object name, class and section are indexed too, so results can be limited by them. Objects are found only by users who may read them: the search checks `content/read` like a fetch (Sections, roles and

policies).

When content is indexed

At publishing, unless `[SearchSettings] DelayedIndexing` is `enabled` (or `classbased` with `DelayedIndexingClassList[]`). Then publishing only queues the object, and the `indexcontent.php` cronjob of the default part indexes it later. Publishing large objects is faster that way, but **a site with delayed indexing depends on cron**: without it new content never appears in search (Delayed indexing and cron, Cronjobs).

The search page

`/content/search` is the search view; `/content/advancedsearch` the advanced form. The result template is `content/search.tpl` of your design chain: override it to match the site, keep the field name `SearchText`, and page with the shipped navigator (Search results template). Limits of the built-in engine are settings in `[SearchSettings]`:

Key	Shipped	Meaning
`MaximumSearchLimit`	`30`	the most results a request may ask for
`AllowEmptySearch`	`disabled`	empty searches are refused
`EnableWildcard`	`false`	wildcard search, which can be slow
`LogSearchStats`	`disabled`	statistics of every phrase typed; they can hold personal data

Searching from templates

Template

```
{def $result = fetch( 'content', 'search', hash(
        'text', $search_text,
        'subtree_array', array( $node.node_id ),
        'limit', 10,
        'offset', $view_parameters.offset ) )}
{foreach $result.SearchResult as $hit}
    {node_view_gui content_node=$hit view='line'}
{/foreach}
<p>{$result.SearchCount} results</p>
{undef $result}
```

The parameters of the `search` fetch (`kernel/content/function_definition.php`): `text`, `subtree_array`, `offset`, `limit`, `publish_timestamp`, `publish_date`, `section_id`, `class_id`, `class_attribute_id`, `ignore_visibility`, `limitation` and `sort_by`; `class_id` takes numeric class IDs. Use it for a search box limited to one part of the site; for lists without a search text, a `list` or `tree` fetch is cheaper (Fetching content in templates).

Searching from PHP

`eZSearch::search( $text, $params )` runs the configured engine, so the same code works with the built-in engine and with eZ Find. Check the parameters your engine accepts in its class before relying on them (`kernel/classes/ezsearch.php`, the engine class named by `SearchEngine`).

Rebuilding the index

After an import, a database move, or when results are wrong:

Shell

```
php bin/php/updatesearchindex.php --clean --allow-root-user
```

`--clean` empties the index first. Run it as the site's user, on a large site in a quiet hour. With eZ Find, reindex with the extension's own tools.

Choosing for a project

Need	Choice
A search box on a small or medium site	the built-in engine
Relevance tuning, facets, spelling suggestions, many languages, large volumes	Solr through eZ Find, after checking the versions it needs
Lists filtered by fields, without a search text	fetches, not search
Results sorted best match first in the editor's dialogs	with eZ Find, `ezfind.ini [SearchSettings] SortBy[]` (The order of search results)

Related pages

- Searching: the editor's view
- Search statistics
- Search course for trainers
- 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.