Docs

Content Project

Where knowledge becomes product

Anonymous · 09/09/2026 · #php

Table of Contents

Content Project 3
- ***Introduction*** 4
- ***Content Frontmatter*** 6
- ***Academy Plugin*** 9
- ***Blog Plugin*** 13
- ***Docs Plugin*** 16
- ***FAQ Plugin*** 18
- ***Pages Plugin*** 20
- ***PDF and Markdown Export*** 22
- ***API Plugin*** 24
- ***Sitemap Plugin*** 26
- ***MCP Plugin*** 27
- ***Search Plugin*** 30
- ***Storage Plugin*** 33
- ***Content Cache*** 35

Content Project

Where knowledge becomes product

Anonymous · 09/09/2026 · #php

Where knowledge becomes product

Last updated on 09/09/2026

Introduction

What derafu/content is and how its plugins fit together.

Anonymous · 09/09/2026

Where knowledge becomes product

Your content — documentation, FAQs, courses, blogs — is not just text. It's a valuable asset that can be packaged, reused, and leveraged like a digital product.

`derafu/content` is a PHP library that turns a directory of Markdown files into a content website: pages, navigation, tags, search and an interface AI agents can call directly. Every plugin adds one content type or one capability on top of the same core.

Two kinds of configuration

Every plugin has, potentially, two independent configuration surfaces, and it is important not to confuse them:

1. **Plugin configuration**, in the website's `services.yaml`, under `derafu.content.config.plugins.<name>`. This is set once by whoever runs the website (which directory to scan, which URL to call, whether a feature is enabled at all).
2. **Content frontmatter**, the YAML block at the top of each Markdown file (`---title: ......`). This is set by whoever writes a specific piece of content (its title, tags, whether it's a draft, etc.).

Both are documented per plugin, but the frontmatter fields are mostly the same across every content type — see Content frontmatter for the fields shared by all of them. Each plugin's page only documents what it adds or overrides on top of that shared set.

Content hierarchy

Content types that support nesting (Docs, Academy, FAQ, Pages) build their hierarchy from the filesystem: a subdirectory represents a level of nesting, and it needs a file with the *same name* as the directory to represent that level itself. For example:

```
docs/
  facturacion.md              # Represents the "facturacion" section itself.
  facturacion/
```

```
    anular-dte.md              # A child of that section.
```

Without `facturacion.md`, the loader has no content item to attach `facturacion/anular-dte.md` to, and the whole subdirectory is silently skipped — it will not show up in listings, search, or the API export. This is the single most common cause of “my content doesn’t show up.”

Missing content is a 404, not a 500

Requesting a URI that doesn’t exist (or a draft outside a local environment) throws `Derafu\Content\Exception\ContentNotFoundException`, which is mapped to **404 Not Found** — across every content plugin (Academy, Blog, Docs, FAQ, Pages) and Storage attachments alike, since they all resolve through the same `ContentRegistryInterface::get()`. A draft that isn’t allowed is treated the same way on purpose, so a 403 doesn’t reveal that it exists.

Plugins

- **Academy**: Course management (course → module → lesson hierarchy).
- **API**: Bulk JSON export of the content, meant for external indexing/RAG pipelines.
- **Blog**: Blog management.
- **Docs**: Documentation management.
- **FAQ**: FAQ management.
- **MCP**: Exposes the content as an MCP (Model Context Protocol) server, so AI agents (Claude Code, Claude Desktop, Cursor, etc.) can search and fetch it directly as tools instead of relying on stale training data.
- **Pages**: Standalone pages management, rendered under a `/pages` prefix by default so it doesn’t conflict with hand-authored flat pages.
- **Search**: Semantic search engine integration, with an optional LLM-based conversational answer (“ask”) grounded on the indexed content.
- **Sitemap**: XML sitemap of every indexable content item, for search engines.
- **Storage**: Attachment storage and download management.

Every content item can also be rendered as HTML, Markdown or PDF, on top of the JSON used by the API and MCP plugins — see PDF and Markdown export for what those two add beyond the raw content (video/quiz rendering, bundling a whole section into one file, download buttons on the HTML view).

Last updated on 09/09/2026

Content Frontmatter

The YAML frontmatter fields shared by every content type (Academy, Blog, Docs, FAQ, Pages).

Anonymous · 09/09/2026

Content frontmatter

Every Markdown content file starts with a YAML frontmatter block:

```
---
title: "How to void a DTE"
description: "Steps to void an electronic tax document"
tags: ["billing", "chile"]
draft: false
---

The body of the content, in Markdown, starts here.
```

These fields are defined once, in `AbstractContentItem`, and apply to **every** content type (Academy lessons/modules/courses, Blog posts, Docs, FAQ questions, Pages). A plugin's own page only lists fields it adds or overrides on top of this shared set — if a plugin's page doesn't mention a field from this table, it behaves exactly as described here.

Unknown keys are not rejected: the frontmatter schema allows undefined keys, so a plugin (or a Twig template) can read a custom field with `item.metadata('my_field')` without declaring it anywhere first.

Identity and SEO

Field	Type	Default	Description
`title`	string	file name	Title of the content.
`description`	string	auto (see `preview` below)	Meta description and card/preview text. `summary` is a deprecated alias.
`keywords`	array of strings	`[]`	Extra keywords added to the `<meta name="keywords">` tag, on top of the tags.
`image`	string	none	Absolute or relative URL used for `og:image` and card previews.
`video`	string	none	Video URL. YouTube "watch" URLs are automatically rewritten to embed URLs.

Field	Type	Default	Description
`slug`	string	slugified file name	Overrides the URL segment used for this item.
`tags`	array of strings	`[]`	Tags shown on the item and used to filter listings (`/type/tags/{tag}`).
`authors`	string, or array of strings/objects (`{name, slug}`)	`Anonymous`	`author` (singular) is a deprecated alias.

Publishing

Field	Type	Default	Description
`draft`	bool	`false`	Drafts are only visible when `APP_ENV=local` or the request host is `localhost`; hidden otherwise.
`unlisted`	bool	`false`	Still reachable by direct URL, but excluded from listings/tag pages unless explicitly filtered by `id`/`uri`.
`date`	string or timestamp	`YYYY-MM-DD-` prefix in the file name, or file creation time	Publish date. `created` is a deprecated alias.
`last_update`	string or timestamp	file modification time	Shown as “last updated on”.
`deprecated`	bool, string or timestamp	`false`	`true` uses the file’s modification time; a string/timestamp sets a specific deprecation date.
`indexable`	bool	`!draft && !unlisted && !deprecated`, and at least 100 characters of body	Whether the item should be considered for the `/api/content.json` export and, from there, external indexing (Qdrant, etc.).
`searchable`	bool	`!draft && !unlisted && !deprecated`	Whether the item should be considered a candidate for the site’s own search (`search` plugin) — independent of `indexable`.
`time`	int (minutes)	auto-estimated from word count (200 wpm)	Reading time. Set it explicitly when the estimate is off (code-heavy pages, etc.).

Sidebar and table of contents

Field	Type	Default	Description
`pagination_label`	string	`title`	Label used in the previous/next pagination links.
`sidebar_label`	string	`title`	Label used in the sidebar, which can differ from the page title.
`sidebar_position`	int	descending by `date` (newest first)	`order` is a deprecated alias. Lower sorts first.
`sidebar_class_name`	string	none	Extra CSS class added to the sidebar entry.
`sidebar_custom_props`	array	`[]`	Arbitrary data made available to the sidebar template.
`hide_title`	bool	`false`	Hides the `<h1>` rendered from `title`.
`hide_table_of_contents`	bool	`false` (`true` for Blog and FAQ)	Hides the in-page table of contents.
`toc_min_heading_level`	int (2-6)	`2`	Minimum heading level included in the table of contents.
`toc_max_heading_level`	int (2-6)	`6`	Maximum heading level included in the table of contents.

Authoring helpers

Field	Type	Default	Description
`has_twig`	bool	auto-detected (`<twig:` in the body)	Whether the Markdown body should be rendered as Twig before being rendered as Markdown, so it can use Twig components.

Content is organized hierarchically on the filesystem for the plugins that support nesting — see Content hierarchy for the “one file per directory level” rule that trips people up most often.

Last updated on 09/09/2026

Academy Plugin

Course management, with a course → module → lesson hierarchy.

Anonymous · 09/09/2026

Academy plugin

Manages online courses with a three-level hierarchy: **course → module → lesson**. Each level is a content item on its own (with its own title, description, tags, etc.), built from the filesystem structure — see Content hierarchy.

```
resources/content/academy/
  getting-started.md                    # Course.
  getting-started/
    introduction.md                     # Module.
    introduction/
      what-is-this.md                   # Lesson.
      how-it-works.md                   # Lesson.
```

Configuration (`services.yaml`)

Enabled under `derafu.content.config.plugins.academy`:

```
parameters:
    derafu.content.config:
        plugins:
            academy:
              path: 'resources/content/academy'
              academyTitle: 'Academy'
              academyDescription: 'Do you want to learn about a topic? Start a course
with us!'
```

Option	Type	Default	Description
`path`	string	`resources/content/academy`	Directory scanned for course content, relative to the website root.
`academyTitle`	string	`Academy`	Title used for the academy's own SEO metadata.

Option	Type	Default	Description
`academyDescription`	string	`Do you want to learn about a topic? Start a course with us!`	Description used for the academy's own SEO metadata.
`include`	array of glob patterns	`['**.{markdown,md}']`	Files considered content, relative to `path`.
`exclude`	array of glob patterns	`[]`	Files excluded even if matched by `include`.
`showReadingTime`	bool	`true`	Whether to show the estimated reading time.
`showLastUpdateAuthor`	bool	`true`	Whether to show who last updated the lesson.
`showLastUpdateTime`	bool	`true`	Whether to show when the lesson was last updated.
`tags`	array of strings, or string	`[]`	Predefined tags for the academy.
`onInlineTags`	`ignore|log|warn|throw`	`warn`	What to do when a lesson uses a tag that isn't in the predefined `tags` list.

Content frontmatter

Courses and modules use exactly the generic content frontmatter, nothing added.

Lessons add one field:

Field	Type	Default	Description
`test`	string	none	A reference to a JSON quiz attachment: either `?attachment=<filename>`, or a literal path ending in `/_attachments/<filename>` (see Storage) — both resolve to the same local attachment. Parsed into a structured test, shown as an interactive quiz on the lesson's own page, and rendered in full in the PDF and Markdown exports. Also switches the lesson's sidebar icon. A `test` value that resolves to neither form is passed through as-is to the quiz widget instead, unparsed.

`time` on a course or module is not read from its own frontmatter — it is always the sum of its lessons' `time` (explicit or estimated).

Quiz JSON format

```
{
    "title": "Introduction quiz",
    "description": "Check what you remember from this lesson.",
    "questions": [
        {
            "type": "multiple_choice",
            "text": "Which of these is correct?",
            "options": [
                { "text": "Option A", "is_correct": true },
                { "text": "Option B", "is_correct": false }
            ],
            "allow_multiple": false,
            "explanation": "Option A is correct because..."
        },
        {
            "type": "true_false",
            "text": "This statement is true.",
            "answer": true,
            "explanation": "..."
        }
    ]
}
```

Field	Type	Description
`title`	string	Title of the test.
`description`	string	Optional description.
`questions[].type`	`multiple_choice`|`true_false`	Question type. A `true_false` question needs no `options` — its two options ("True"/"False") are generated from `answer`.

Field	Type	Description
`questions[].options[].is_correct`	bool	Whether this option is a correct answer. More than one can be `true` when `allow_multiple` is `true`.
`questions[].explanation`	string	Shown in the PDF/Markdown exports' answer key, and in the lesson's own interactive quiz when an answer is marked incorrect.

Last updated on 09/09/2026

Blog Plugin

Blog management, with archive, tags and an RSS feed.

Anonymous · 09/09/2026

Blog plugin

Manages blog posts: listing, tag pages (`/blog/tags/{tag}`), a date archive (`/blog/archive/{archive}`) and an RSS feed (`/blog/rss.xml`).

Configuration (`services.yaml`)

Enabled under `derafu.content.config.plugins.blog`:

```
parameters:
    derafu.content.config:
        plugins:
            blog:
              path: 'resources/content/blog'
              blogTitle: 'Blog'
              postsPerPage: 10
```

Option	Type	Default	Description
`path`	string	`resources/content/blog`	Directory scanned for posts, relative to the website root.
`blogTitle`	string	`Blog`	Title used for the blog's own SEO metadata.
`blogDescription`	string	`Thoughts, stories, and the latest from our world.`	Description used for the blog's own SEO metadata.
`blogSidebarCount`	int	`5`	Number of recent posts shown in the sidebar.
`blogSidebarTitle`	string	`Recent posts`	Title of the recent-posts sidebar.
`include`	array of glob patterns	`['**.{markdown,md}']`	Files considered content, relative to `path`.

Option	Type	Default	Description
`exclude`	array of glob patterns	`[]`	Files excluded even if matched by `include`.
`postsPerPage`	int	`10`	Posts per page in the listing.
`showReadingTime`	bool	`true`	Whether to show the estimated reading time.
`feedOptions`	array	see below	RSS feed configuration (`/blog/rss.xml`).
`showLastUpdateAuthor`	bool	`true`	Whether to show who last updated the post.
`showLastUpdateTime`	bool	`true`	Whether to show when the post was last updated.
`tags`	array of strings, or string	`[]`	Predefined tags for the blog.
`onInlineTags`	`ignore|log|warn|throw`	`warn`	What to do when a post uses a tag that isn't in the predefined `tags` list.

`feedOptions`

Option	Type	Default	Description
`limit`	int	`20`	Number of posts included in the feed.
`title`	string	`blogTitle`	Overrides the feed title.
`description`	string	`blogDescription`	Overrides the feed description.
`copyright`	string	none	Copyright line of the feed.
`language`	string	none	Feed language.
`sortPosts`	`descending|ascending`	`descending`	Sort order of the posts in the feed.

Content frontmatter

Blog posts use the generic content frontmatter, with one override:

Field	Type	Default	Description
`hide_table_of_contents`	bool	`true`	Overrides the generic default of `false` — blog posts hide the table of contents unless explicitly re-enabled.

Last updated on 09/09/2026

Docs Plugin

Documentation management, with a sidebar and nested sections.

Anonymous · 09/09/2026

Docs plugin

Manages documentation pages, nested arbitrarily deep — see Content hierarchy. Renders as HTML, Markdown or PDF depending on the requested format (`Accept` header, or `.md`/`.pdf` suffix) — see PDF and Markdown export for what the last two add, including bundling a whole section into one file with `?full=1`.

Configuration (`services.yaml`)

Enabled under `derafu.content.config.plugins.docs`:

```
parameters:
    derafu.content.config:
        plugins:
            docs:
              path: 'resources/content/docs'
              sidebarDepth: 5
```

Option	Type	Default	Description
`path`	string	`resources/content/docs`	Directory scanned for docs, relative to the website root.
`include`	array of glob patterns	`['**.{markdown,md}']`	Files considered content, relative to `path`.
`exclude`	array of glob patterns	`[]`	Files excluded even if matched by `include`.
`sidebarPath`	bool	`true`	Whether to show the sidebar, auto-generated from the content hierarchy. There is no support for a custom, manually curated sidebar file — sort/label docs via `sidebar_position`/`sidebar_label` in their frontmatter instead.
`sidebarCollapsible`	bool	`true`	Whether sidebar categories can be collapsed.
`sidebarCollapsed`	bool	`true`	Whether sidebar categories start collapsed.

Option	Type	Default	Description
`sidebarDepth`	int	`5`	Maximum nesting depth shown in the sidebar.
`showLastUpdateAuthor`	bool	`true`	Whether to show who last updated the doc.
`showLastUpdateTime`	bool	`true`	Whether to show when the doc was last updated.
`breadcrumbs`	bool	`true`	Whether to show breadcrumbs above the doc.
`tags`	array of strings, or string	`[]`	Predefined tags for the docs.
`onInlineTags`	`ignore|log|warn|throw`	`warn`	What to do when a doc uses a tag that isn't in the predefined `tags` list.

Content frontmatter

Docs use the generic content frontmatter, plus a few fields read directly by the docs template (not part of the shared schema, but supported the same way via arbitrary metadata):

Field	Type	Default	Description
`show_title`	bool	`false`	Renders `# {title}` at the top of the body — useful together with `hide_title: true` if you want the title placed differently than the default heading.
`show_description`	bool	`false`	Renders the `description` right below the title inside the body.
`iframe`	string	none	URL embedded as a full-width `<iframe>` below the content.
`openapi`	string	none	URL of an OpenAPI/Swagger spec; renders a full Swagger UI below the content instead of (or in addition to) the Markdown body.
`show_source`	bool	`false`	Shows the raw source of the file (Markdown or Twig) in a code block below the content.
`show_children`	bool	`false`	Renders a card grid linking to this doc's direct children (title + description), useful for section landing pages.

`hide_table_of_contents` keeps the generic default (`false`) for this plugin.

Last updated on 09/09/2026

FAQ Plugin

FAQ management, with nested sections and a sidebar.

Anonymous · 09/09/2026

FAQ plugin

Manages frequently asked questions, nested arbitrarily deep — see Content hierarchy. Structurally very similar to Docs, just under `/faq` instead of `/docs`.

Configuration (`services.yaml`)

Enabled under `derafu.content.config.plugins.faq`:

```
parameters:
    derafu.content.config:
        plugins:
            faq: ~
```

Option	Type	Default	Description
`path`	string	`resources/content/faq`	Directory scanned for questions, relative to the website root.
`include`	array of glob patterns	`['**.{markdown,md}']`	Files considered content, relative to `path`.
`exclude`	array of glob patterns	`[]`	Files excluded even if matched by `include`.
`sidebarPath`	bool	`true`	Whether to show the sidebar, auto-generated from the content hierarchy. There is no support for a custom, manually curated sidebar file — sort/label questions via `sidebar_position`/`sidebar_label` in their frontmatter instead.
`sidebarCollapsible`	bool	`true`	Whether sidebar categories can be collapsed.
`sidebarCollapsed`	bool	`true`	Whether sidebar categories start collapsed.
`sidebarDepth`	int	`5`	Maximum nesting depth shown in the sidebar.
`showLastUpdateAuthor`	bool	`false`	Whether to show who last updated the question.

Option	Type	Default	Description
`showLastUpdateTime`	bool	`true`	Whether to show when the question was last updated.
`breadcrumbs`	bool	`true`	Whether to show breadcrumbs above the question.
`tags`	array of strings, or string	`[]`	Predefined tags for the FAQ.
`onInlineTags`	`ignore`|`log`|`warn`|`throw`	`warn`	What to do when a question uses a tag that isn't in the predefined `tags` list.

Content frontmatter

FAQ questions use the generic content frontmatter, with one override and one addition:

Field	Type	Default	Description
`hide_table_of_contents`	bool	`true`	Overrides the generic default of `false` — questions hide the table of contents unless explicitly re-enabled.
`show_children`	bool	`false`	Renders a card grid linking to this question's direct children (title + description), useful for section landing pages.

Last updated on 09/09/2026

Pages Plugin

Standalone pages management, rendered as HTML, Markdown, PDF or JSON.

Anonymous · 09/09/2026

Pages plugin

Manages standalone pages (an “about us”, a landing page, changelog, etc.), including `.html.twig` files in addition to Markdown. Same content model and format negotiation (HTML/Markdown/PDF/JSON) as Docs/FAQ, just without a listing/tag view — pages are meant to be linked directly, not browsed.

Route and the `/pages` prefix

Route	Path	Description
`pages_page`	`GET /pages/{uri}`	Shows a page. Format negotiated the same way as every other content plugin (`Accept` header, or `.md`/`.pdf`/`.json` suffix).

The route is registered under `/pages` **on purpose**, not at the root (`/about` instead of `/pages/about`). The `{page:.+}` pattern is a catch-all, and the router tries routes with parameters *before* it tries file-based routes (`Derafu\Routing\Parser\FileSystemParser`, used for standalone Twig pages placed directly in `templates/pages/` by the website itself, with no content model). A catch-all with no prefix would shadow every one of those file-based pages: the router would never even get to ask the file-system parser about a URI this route already claims, since it checks parsers in order and stops at the first one that matches.

We verified this in practice: with the plugin enabled and content under `resources/content/pages/`, every existing hand-written route (static ones, and file-based ones from `templates/pages/`) kept working exactly as before — only URIs actually starting with `/pages/` are affected.

If a website wants pages at the root anyway, it can register its own route pointing at the same handler instead of importing this one:

```
pages_page:
  path: /{page:.+}
  handler: 'Derafu\Content\Plugin\Pages\PagesController::show'
```

Registering that **before** any route that relies on `FileSystemParser`’s auto-discovery turns this trade-off off for that whole website: from then on, every flat page must be a content item (title/tags/draft/etc. via frontmatter), not a hand-authored Twig template dropped into `templates/pages/`. Pick one mechanism per website, not both, once this is unprefixed.

Configuration (`services.yaml`)

Enabled under `derafu.content.config.plugins.pages`:

```
parameters:
    derafu.content.config:
        plugins:
            pages:
              path: 'resources/content/pages'
```

Option	Type	Default	Description
`path`	string	`resources/content/pages`	Directory scanned for pages, relative to the website root.
`include`	array of glob patterns	`['**.{markdown,md,html.twig}']`	Files considered content, relative to `path`. Unlike other plugins, this also matches `.html.twig` files.
`exclude`	array of glob patterns	`[]`	Files excluded even if matched by `include`.
`showLastUpdateAuthor`	bool	`false`	Whether to show who last updated the page.
`showLastUpdateTime`	bool	`false`	Whether to show when the page was last updated.

Content frontmatter

Pages use exactly the generic content frontmatter, nothing added.

Last updated on 09/09/2026

PDF and Markdown Export

What the PDF and Markdown exports add on top of the raw content, and how to bundle a whole section into one file.

Anonymous · 09/09/2026

PDF and Markdown export

Every content item (Academy, Blog, Docs, FAQ, Pages) renders as HTML, Markdown, PDF or JSON — format negotiated via the `Accept` header, or a `.md`/`.pdf`/`.json` suffix on the URL. HTML is the normal browsing experience and JSON is the same shape used by API/MCP; this page covers what is specific to the PDF and Markdown exports.

Root-relative URLs are resolved to absolute ones

A content body written for the website itself references images and internal links with root-relative paths (`/img/foo.png`, `/docs/other-page`). Neither export has a "current page" to resolve a root-relative path against, so both make the body self-contained wherever it ends up — an LLM's context window, someone else's Markdown viewer, a PDF saved to disk:

- **Markdown**: both images and internal links are rewritten to absolute URLs.
- **PDF**: only links are rewritten. Images are left root-relative on purpose — the PDF engine resolves them from the local filesystem instead of fetching them over HTTP, which is slower and, on a single-worker server, can tie up the whole process waiting on a request back to itself.

Video and quiz are shown, not just left in the frontmatter

A frontmatter `video` (see Content frontmatter) is a plain URL in YAML that neither export would otherwise surface anywhere a reader (or an LLM) would notice it:

- **PDF**: a clickable YouTube thumbnail linking to the watch page, or a plain link for a non-YouTube URL.
- **Markdown**: the same thumbnail and link, as a Markdown image wrapped in a link.

An Academy lesson's `test` (quiz) is rendered in full — every option, not only the correct one, plus the explanation, since the wrong options and the explanation are real information (useful to know what an answer is commonly confused with, or why it's wrong):

- **PDF**: the questions with checkbox-style options first (to answer on paper), then a dedicated "Answers" section on its own page, with the correct option(s) checked and the explanation for each question.

- **Markdown**: each question as a GFM task list (`- [x]`/`- [ ]`, the correct option(s) checked), followed by the explanation as a blockquote.

Bundling a whole section: `?full=1`

Content types with children — a Docs or FAQ section, an Academy course or module — accept `?full=1` on their `.pdf`/`.md` URL to bundle the whole subtree into a single file, instead of just the current item with a short list of links to its children:

- **PDF**: a cover page, a real table of contents with page numbers, then every descendant on its own page (Docs/FAQ children recursively; an Academy course's modules and, nested under each, its lessons).
- **Markdown**: every descendant's title and full body appended after the current item's, with heading levels reflecting how deep it is nested.

`?full=1` on an item with no children is a no-op: it falls through to the regular, single-item export.

Download buttons on the HTML view

Every content page shows small, always-visible buttons for its PDF and Markdown exports, next to the tags/last-updated line — plain buttons, not a dropdown, so there is nothing to discover before they can be used:

- **PDF** opens inline: browsers render it in their own viewer, which already has its own save/print controls.
- **Markdown** downloads instead of opening inline: a browser has no native Markdown renderer, so opening it in a tab just shows unstyled raw text. A "Copy" button next to it copies the Markdown source straight to the clipboard — handy for pasting into an LLM — and confirms with a Notyf toast when the site has it loaded, falling back to a brief change of the button's own text otherwise.

"Full" (and "Full Copy") variants of these buttons appear next to the regular ones whenever the item actually has children to bundle.

Last updated on 09/09/2026

API Plugin

Bulk JSON export of the content, meant for external indexing/RAG pipelines.

Anonymous · 09/09/2026

API plugin

Exposes a single endpoint, `GET /api/content.json`, that returns every **indexable** item across every content plugin (Academy, Blog, Docs, FAQ, Pages) as a flat JSON list, via `ContentService::allContent()` — the same aggregation the Sitemap plugin uses. This is the feed an external pipeline (e.g. a tokenizer that loads embeddings into Qdrant) is meant to consume, and it is the source of truth external tools index against — not a page-by-page crawl of the website.


```
{
    "meta": {
        "count": 424,
        "url": "https://example.com/",
        "generated": "2026-08-17 16:52:39"
    },
    "data": [
        {
            "id": "docs_doc_facturacion_anular-dte",
            "checksum": "b256587c...",
            "type": "docs",
            "category": "doc",
            "uri": "facturacion/anular-dte",
            "link": "https://example.com/docs/facturacion/anular-dte.json",
            "image": null,
            "title": "Anular un DTE",
            "description": "Pasos para anular un Documento Tributario Electrónico",
            "authors": [{ "name": "Anonymous", "slug": "anonymous" }],
            "tags": [{ "name": "facturacion", "slug": "facturacion", "count": 12 }],
            "date": "2026-03-12",
            "last_update": "2026-03-12",
            "time": 2
        }
    ]
}
```


Items with an empty body (`data()`), or that are not `indexable` (see generic frontmatter), are skipped.

Configuration (`services.yaml`)

None. Enabling the plugin (`api: ~`) is all that's needed — it has no options of its own, it only reads whatever the Academy/Blog/Docs/FAQ/Pages plugins already loaded.

```
parameters:
    derafu.content.config:
        plugins:
            api: ~
```

Filtering

The endpoint accepts the same filters as `ContentRegistry::filter()` as query parameters — for example `/api/content.json?type=docs&tag=facturacion&limit=20&page=1`. Useful keys: `type`, `category`, `tag`, `author`, `search`, `year`+`month`, `indexable`, `searchable`, `limit`, `page`.

Content frontmatter

Not applicable — this plugin has no content items of its own, it aggregates the ones from the plugins that do.

Last updated on 09/09/2026

Sitemap Plugin

XML sitemap of every indexable content item, for search engines.

Anonymous · 09/09/2026

Sitemap plugin

Exposes a single endpoint, `GET /sitemap.xml`, listing every **indexable** item across every content plugin (Academy, Blog, Docs, FAQ, Pages) — for search engine crawlers, not for AI agents (that's MCP) nor for an indexing pipeline (that's API). Deliberately does not emit `<changefreq>`/`<priority>`: modern crawlers (Google included) ignore both, so there is nothing to configure there.

```
<?xml version="1.0" encoding="UTF-8" ?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>https://example.com/docs/facturacion/anular-dte</loc>
        <lastmod>2026-03-12T17:11:58+00:00</lastmod>
    </url>
</urlset>
```

Items with `indexable: false` (see generic frontmatter) are skipped — the same rule the API plugin's export uses.

Configuration (`services.yaml`)

None. Enabling the plugin (`sitemap: ~`) is all that's needed.

```
parameters:
    derafu.content.config:
        plugins:
            sitemap: ~
```

Content frontmatter

Not applicable — this plugin has no content items of its own, it aggregates the ones from the plugins that do, the same way the API plugin does (via `ContentService::allContent()`).

MCP Plugin

Exposes the content as an MCP (Model Context Protocol) server for AI agents.

Anonymous · 09/09/2026

MCP plugin

Exposes the content of the website as an MCP (Model Context Protocol) server, so AI agents (Claude Code, Claude Desktop, Cursor, claude.ai connectors, etc.) can search and fetch it directly as tools during their own conversation — instead of relying on stale training data or scraping HTML.

This is a different consumption channel than the API plugin: the API plugin is a one-shot bulk export meant for an indexing pipeline (Qdrant, etc.); the MCP plugin is a live, callable interface for any MCP-capable agent.

Endpoint

A single route, `POST /api/mcp` (also accepts `DELETE` to end a session, and `OPTIONS` for CORS preflight). It deliberately has **no `.json` suffix**: unlike every other endpoint in this package, its response is not always JSON — it can also be `text/event-stream` (SSE) when the protocol needs to keep the connection open mid-call. The MCP protocol negotiates that per-request via the `Accept` header, not via the URL, and every real MCP client already expects a plain endpoint URL with no extension.

It speaks JSON-RPC 2.0 over the “Streamable HTTP” transport of the official PHP MCP SDK (`mcp/sdk`), synchronously — there is no event loop involved, it fits the same request/response model as every other controller in this package.

Configuration (`services.yaml`)

Enabled under `derafu.content.config.plugins.mcp`:

```
parameters:
    derafu.content.config:
        plugins:
            mcp:
              ask:
                enabled: true
```

Option	Type	Default	Description
`server_name`	string	`derafu-content`	Name announced to MCP clients during the `initialize` handshake.
`server_version`	string	`1.0.0`	Version announced to MCP clients during the `initialize` handshake.
`session_path`	string	a subdirectory of `sys_get_temp_dir()`	Directory where MCP sessions are persisted on disk between requests (a session is created on `initialize` and referenced by later calls via the `Mcp-Session-Id` header; since each request builds a new server instance, this cannot live in memory). Point it at something under the website's `var/` directory for a more durable/cleanable location.
`session_ttl`	int	`3600`	Time to live, in seconds, of a persisted session.
`ask.enabled`	bool	`false`	Whether the `ask` tool (LLM-backed conversational answers) is registered. See Tools below.

This plugin does not handle authentication nor rate limiting — that is not its responsibility. If the endpoint needs to be protected, do it at the HTTP stack level (e.g. with the middlewares of `derafu/http`), before this controller is reached.

The `mcp/sdk` dependency

`mcp/sdk` is a `require-dev` dependency (with a `suggest` entry) of `derafu/content`, not a hard `require`. If you enable this plugin, add it explicitly to **your own** `composer.json`:

```
{
    "require": {
        "mcp/sdk": "^0.7"
    }
}
```

If you don't enable the plugin, nothing needs it: hitting `/api/mcp` without it installed and without the plugin configured fails with a normal "plugin not found" error, exactly like hitting `/api/search.json` without the `search` plugin configured — it does not affect any other endpoint of the website.

Tools

Tool	Always registered?	Description
`search_content`	yes	Semantic search across the indexed content, delegating to the Search plugin's engine (Qdrant, or whatever is configured there). Accepts an optional `source` (`academy`, `blog`, `docs`, `faq`, `pages`, or `all`) to scope the search; omitting it or passing `all` searches every source at once.
`get_content`	yes	Fetch a single item by source and URI: full Markdown body plus metadata (title, tags, dates, authors).
`list_content`	yes	Browse/filter the items of a source (academy, blog, docs, faq, pages), optionally by tag, category or free-text search.
`list_tags`	yes	List the tags used in a source, with how many items use each one.
`ask`	only if `ask.enabled: true`	Ask a natural-language question and get a conversational answer from the LLM configured in the Search plugin, grounded on the indexed content.

`ask` is opt-in because, unlike the other four tools (which only ever read from the indexed content), it depends entirely on the quality of whatever LLM backend answers it — a bad or failed answer there makes the whole MCP server look unreliable, even though the rest of the tools never touch an LLM at all.

Content frontmatter

Not applicable — this plugin has no content items of its own, it reuses whichever content plugins (Academy, Blog, Docs, FAQ, Pages) are already enabled.

Last updated on 09/09/2026

Search Plugin

Search engine integration, with an optional LLM-based conversational answer.

Anonymous · 09/09/2026

Search plugin

Provides a search page and API that proxy to an **external** search engine — this plugin does not index anything itself, it queries whatever semantic/full-text search service you point it at (in practice, a service backed by Qdrant, fed by the API plugin's export). It optionally also proxies to an LLM to answer questions conversationally, grounded on that same indexed content.

Routes

Route	Path	Description
`search`	`GET /search`	Search page (HTML).
`search_api`	`GET /api/search.json?q=...`	Search results as JSON.
`search_llm_query`	`GET /api/search/llm.json?q=...`	Conversational answer from the LLM (see LLM backend below).

Configuration (`services.yaml`)

Enabled under `derafu.content.config.plugins.search`:

```
parameters:
    derafu.content.config:
        plugins:
            search:
              url:
'https://search.example.com/api/search?collection=%s&base_url=%s&text=%s'
              collection: 'my-site'
              base_url: 'https://example.com'
              llm_url: 'https://api.openai.com'
              llm_model: 'gpt-4o-mini'
              llm_api_key: '%env(OPENAI_API_KEY)%'
```

Option	Type	Default	Description
`url`	string	*(required)*	`sprintf()` template for the search engine's URL. See URL template below.

Option	Type	Default	Description
`collection`	string	none	Collection/index identifier, URL-encoded and injected into `url`.
`base_url`	string	none	Base URL of the website, URL-encoded and injected into `url` (useful when the search backend serves more than one site).
`llm_url`	string	none	Base URL of the LLM backend. Leave unset to disable the `ask` tool (MCP) and make `search_llm_query` fail.
`llm_model`	string	none	Model name sent to the LLM backend. **Required if `llm_url` is set** — there is no generic default, since no single model name makes sense across every provider. `llm()` throws immediately with a clear message if `llm_url` is configured without it, instead of silently sending an empty model name to your backend.
`llm_api_key`	string	none	API key sent as `Authorization: Bearer <key>`.
`llm_completions_path`	string	`/v1/chat/completions`	Path of the chat completions endpoint, appended to `llm_url`. See LLM backend.

URL template

`url` is a `sprintf()` template consumed in this order — trailing placeholders can be omitted:

1. `%s` → collection (only if both `collection` and `base_url` are set).
2. `%s` → base URL (only if both `collection` and `base_url` are set).
3. `%s` → the URL-encoded search query (always present, always last).

If `collection` or `base_url` is not configured, `url` is expected to have a single `%s` for the query only.

LLM backend

The `llm_*` options configure a client for **any backend exposing an OpenAI-compatible chat completions endpoint** — the `{model, messages}` request / `choices[0].message.content` response shape that has become a de facto standard. With no code changes, that covers:

- OpenAI itself (`llm_url: https://api.openai.com`, default `llm_completions_path`).
- OpenRouter (`llm_url: https://openrouter.ai/api`, default `llm_completions_path`, `llm_model` like `anthropic/claude-sonnet-4.5`) — itself proxies Claude, Gemini, Llama, etc. through the same shape.

- Anthropic's own OpenAI-compatible endpoint (`llm_url: https://api.anthropic.com`, default `llm_completions_path`).
- Self-hosted Open WebUI — needs `llm_completions_path: /api/chat/completions` instead of the default, since it does not follow the standard path.
- Groq, Together AI, Ollama, Azure OpenAI (`Authorization: Bearer` variant), and most other providers.

A backend that does not speak this protocol at all (a provider's native, non-compatible API) is not supported today, but the client is behind an interface (`Derafu\Content\Plugin\Search\Contract\LlmClientInterface`) precisely so an alternative implementation can be swapped in later without touching `SearchController` or the `ask` MCP tool.

Failures from either the search engine or the LLM backend (unreachable host, non-200 response, unexpected response shape) surface as `SearchUpstreamException`, mapped to **502 Bad Gateway** — not a generic 500 — since the request to this website was fine, it's whatever it depends on that failed. The error message includes whatever detail the upstream returned (`error`/`error.message`/`detail`, depending on the provider).

Content frontmatter

Not applicable — this plugin has no content items of its own; `searchable` (see generic frontmatter) is read by whoever builds the external search index, not by this plugin.

Last updated on 09/09/2026

Storage Plugin

Attachment storage and download management for any content type.

Anonymous · 09/09/2026

Storage plugin

Serves file attachments that live next to a content item — images, PDFs, quiz files for an Academy lesson, etc. — through a single, uniform URL, regardless of which content plugin owns the item.

Attachment convention

An attachment is any file placed in a `_attachments/` subdirectory next to its content file, named after that file (without extension):

```
resources/content/academy/getting-started/introduction/what-is-this.md
resources/content/academy/getting-started/introduction/what-is-this/_attachments/
  cheat-sheet.pdf
  quiz.json
```

`what-is-this.md`'s attachments are exactly the files under `what-is-this/_attachments/`. Reference one from frontmatter as `?attachment=<filename>` — see the `test` field of Academy lessons for a real example.

Route

Route	Path	Description
`content_storage_download`	`GET /{type}/{uri}/_attachments/{attachment}`	Downloads an attachment. `type` is the content type (`docs`, `academy`, etc.), `uri` is the item's URI.

Configuration (`services.yaml`)

None. Enabling the plugin (`storage: ~`) is all that's needed — it has no options of its own, it only serves whatever attachments already exist next to items loaded by the other content plugins.

```
parameters:
```

```
    derafu.content.config:
        plugins:
            storage: ~
```

Content frontmatter

Not applicable — attachments are files on disk, not content items, and are not referenced through frontmatter fields of their own (beyond however a specific plugin points at them, like Academy's `test`).

Last updated on 09/09/2026

Content Cache

What the built-in content cache does and does not speed up, and how to swap its backend.

Anonymous · 09/09/2026

Content cache

Every content plugin that supports nesting (Academy, Blog, Docs, FAQ, Pages) builds its item tree by scanning the filesystem and parsing the YAML frontmatter of every single file — on every request, since this package has no build step and no long-running process. The content cache exists to avoid redoing that scan-and-parse work on every request; it is not a general-purpose page cache.

What it caches

Exactly one thing: the **already-built item tree** of a content registry (the result of `AbstractContentRegistry::all()`) — titles, tags, dates, hierarchy, everything derived from frontmatter. It is keyed by the plugin's `path` + `include` + `exclude` options, so two registries pointed at the same content share one cache entry regardless of which plugin instantiated them first.

This benefits every operation that needs to look at more than one item — listings, tag pages, the sidebar, Sitemap, and API's `allContent()` export — since those are exactly the operations that today pay the "read and parse everything" cost on every request, whether or not the specific page requested needed most of that data.

What it does not cache

To avoid false expectations:

- **Rendered output.** HTML, Markdown, PDF and JSON responses are all still generated fresh on every request — Twig compilation, the `markdown`/`twig` filters, PDF generation. Caching the index does not cache a page.
- **HTTP-level caching.** No `Cache-Control` or `ETag` header is set by any content controller. A reverse proxy or CDN in front of the site would still treat every request as uncacheable unless configured to do otherwise, independently of this feature.
- **Search's external calls.** Queries to the search engine and the LLM backend are never cached — they hit the configured `url`/`llm_url` on every call.
- **MCP tool responses.** Each tool call resolves fresh against the (possibly cached) registries; the MCP layer itself adds no caching of its own.
- **Storage attachment downloads.** Files are streamed from disk on every request.

In short: this makes it cheaper to *know what content exists and what it's about*. It does not make any specific page's response faster to render or serve.

Default: a real filesystem cache, no configuration needed

Caching is on by default, with no service to run. The package's own `content-services.yaml` (imported by every site that enables this package) wires a `Symfony\Component\Cache\Adapter\FilesystemAdapter` under namespace `derafu_content`, 60-second TTL, rooted at `%kernel.cache_dir%` — the exact same `var/cache/<env>/` directory the rest of a `derafu/kernel`-based app already uses (compiled container, etc.). Concretely, that means `var/cache/dev/derafu_content/` in a `dev` environment, `var/cache/prod/derafu_content/` in `prod`, and so on — clearing it is `rm -rf var/cache/<env>/derafu_content`, the same mental model as clearing any other cache in the app, nothing extra to remember.

`Derafu\Content\ContentContext` itself defaults to a `FilesystemAdapter` under the system temp directory if no cache pool is injected at all — that fallback only matters if this package is used standalone, outside the shipped `services.yaml` (e.g. directly in PHP, or in this package's own test suite). Any real site importing `content-services.yaml` gets the `var/cache/<env>/derafu_content` location described above, not the temp directory.

Freshness

Each cache entry expires after 60 seconds. There is no manual invalidation and no filesystem-change detection: editing a Markdown file is not reflected instantly, only once its registry's entry expires (up to 60 seconds later) or the process restarts. This is a deliberate trade-off — a short TTL removes the "rescan everything on every request" cost without needing any invalidation logic — but it does mean content changes are not instantaneous the way they were without caching.

On a multi-worker or multi-server deployment, each worker/server builds and caches its own copy independently unless they share the cache backend (e.g. all pointed at the same Redis instance) — see below.

Disabling the cache: `cache.enabled`

Enabled under `derafu.content.config`:

```
parameters:
    derafu.content.config:
        cache:
            enabled: true
```

Option	Type	Default	Description
`enabled`	bool	`true`	Whether the content cache is used at all. When `false`, `ContentContext::cache()` returns `null` and every registry falls back to `AbstractContentRegistry`'s own `null`-cache path: it rescans and reparses on every request, with no pool involved.

This package reads that value as a plain boolean — it does not know about environments or kernels. Sourcing it from an environment variable, with whatever per-environment default a site wants, is entirely the site's own `services.yaml` concern:

```
parameters:
    derafu.content.config:
        cache:
            enabled: '%env(bool:DERAFU_CONTENT_CACHE_ENABLED)%'
```

`DERAFU_CONTENT_CACHE_ENABLED` then comes from wherever the site already resolves per-environment values (a `.env`/`.env.local` cascade, a `when@dev:` config block using `%kernel.environment%`, etc.) — this package never reads `$_SERVER['APP_ENV']` or the kernel for this decision, only the boolean it's handed.

Using a different cache backend

The cache pool is the `derafu_content.cache` service, typed against `Psr\Cache\CacheItemPoolInterface` (PSR-6) where it's injected into `ContentContext`. A site's own `services.yaml` is loaded after the package's, so redefining that same service ID replaces it outright — no need to touch `ContentContextInterface`'s own definition:

```
services:
    derafu_content.cache:
        class: Symfony\Component\Cache\Adapter\RedisAdapter
        arguments:
            $redis: '@Redis'
            $namespace: 'derafu_content'
            $defaultLifetime: 60
```

This is the right move on a multi-worker or multi-server deployment: with the default `FilesystemAdapter`, each worker/server builds and caches its own copy independently; pointing `derafu_content.cache` at a shared backend (Redis, Memcached) instead means they all share one cache instead of duplicating the work. This is a different concern than `cache.enabled` above: swapping the backend still caches, just somewhere shared; `cache.enabled: false` skips caching altogether regardless of which backend is wired.

Content frontmatter

Not applicable — caching operates on already-loaded items, it adds no frontmatter fields of its own.

Last updated on 09/09/2026