Learning Exponential: the chapters of the Learning book, brought up to date

Last changed 2026-11-04
/exphelp/manual/help-gen-manual-book-learning

Help pages adapted from the free book Learning eZ publish, chapter by chapter, with every step, setting and label checked against Exponential 6.

Learning eZ publish: Building Content Management Solutions is the first book written about the system Exponential continues. It is free documentation: the second edition (2024) is published by 7x under the GNU Free Documentation License (Book-Learning-eZPublish). These pages take its chapters one at a time and rewrite each for Exponential 6: the steps that moved, the settings that were renamed, the examples that no longer work, and the parts that did not change at all, which is most of the concepts.

Every page keeps the book's order of ideas, says plainly what is different now, and links to the current guide for the details. Each ends with the credit and license of the book. Learning from the free books explains how the books relate to this help and how to read the original against Exponential 6.

Chapter 1: Installing

1. Installing, then and now: the files, the database, the virtual host
2. The setup wizard, then and now
3. Installation problems, then and now

Chapter 2: Content management

1. What is content management?
2. The node tree and sections
3. Classes, attributes and objects
4. Content and presentation
5. Roles and policies, step by step
6. Creating a content class
7. The datatypes tour
8. Rich text tags
9. Creating and editing objects
10. Versions, translations and related content
11. Workflows and triggers
12. More functions of the administration
13. Planning an example site: the club portal
14. Users, groups and roles in the example

Chapter 3: Templates

1. Template principles: page layout, content views, attributes
2. Template syntax basics
3. Fetching content in templates
4. Caching in templates
5. Overriding templates
6. Common template tasks

Chapter 4: Inside the core

1. Inside the core: permissions and persistent objects
2. Inside the core: content classes, workflows, notifications, information collection, search

Chapter 5: Extending

1. Extending: why, where and how
2. The building blocks of an extension

Chapter 6: Extension development

1. Extension development practices

Chapter 7: Deploying

1. Deploying: planning the hosting and checking the server

Chapter 9: A standards-compliant site

1. A standards-compliant site: markup, style and structure

The tutorial that builds a complete small site is in Building a site step by step.

Chapter 8 of the book is a case study of one client project and is not adapted; its lessons are in chapters 7 and 9 above.

Source and license

Adapted from *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, published by 7x, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license). Changes: each chapter is split into pages, rewritten for Exponential 6 and checked against the code and the administration.

Installing, then and now: the files, the database, the virtual host

What the Learning book's first chapter had you do by hand (unpack, create a database, write a virtual host, set up images and cron, read the ini files), and what each step is in Exponential 6.

Chapter 1 of the book *Learning eZ Publish* opens with the work every installation needs: get the files, make a database, point the web server at the files, set up image conversion and a regular job, and learn where the settings live. The steps still exist. Most of them are one command today. This page walks the book's order and says what to do now.

> **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.

Unpacking the installation

The book downloaded an archive of about 4 MB, expanded it to about 20 MB and transferred the folders to the web server. Exponential has three ways to get the same files (see Getting started):

- a Git clone of the kernel, with the extensions as clones (the way to build a website you maintain),
- `composer create-project se7enxweb/exponential`,
- and, for single add-ons later, packages installed with `ezpm`.

After either of the first two, the folders the book listed are all there:

Book	Exponential 6
`design`	`design/`: the default templates and styles. Your own go in an extension's `design/` folder, not here.
`index.php`	`index.php`: every page request still enters here.
`kernel`, `lib`	`kernel/`, `lib/`: the code. Do not edit them; see kernel overrides, the last resort.
`runcronjobs.php`	`runcronjobs.php`: still the script a regular job runs (Cronjobs).
`settings`	`settings/`: defaults, plus `override/` and `siteaccess/` for yours.
`var`	`var/`: uploaded files, caches and logs. The web server's user must be able to write here.

New in Exponential: `extension/` (add-ons), `console` (the command line front end, `./console list`) and `vendor/` (Composer packages).

Initializing the database

The book created a MySQL database and a user that may use only that database, with `CREATE DATABASE` and `GRANT ALL PRIVILEGES`, and recommended that user rather than a powerful one, because the password is kept in a settings file in plain text. That advice is unchanged. Two things are new:

- You need no database server to start: the default is SQLite in a file. MySQL or MariaDB, PostgreSQL, MongoDB and Oracle are choices of `exp:install --db=` and of the setup wizard (Databases).
- In `GRANT`, MySQL 8 and MariaDB no longer accept `IDENTIFIED BY` inside it; create the user first with `CREATE USER`, then `GRANT`. The installation book shows the statements.

The virtual host and rewriting

The book explains why a virtual host with `mod_rewrite` is the right setup: all requests go to `index.php`, URLs such as `/node/view/253` become readable addresses like `/contact`, and only that one script is reachable. The two rewrite lines of the book (everything that is not a script, image or style file goes to `index.php`) are what the shipped `.htaccess` does today, in a longer and safer form. With Apache, use the shipped `.htaccess` or put its rules in the host's configuration. To try a site without any web server, Exponential brings its own, Velocity (Serving the site).

Running the site in a subfolder, which the book allows, is not recommended any more: the siteaccess and URL alias logic is built for a site at the root of a host name.

Image settings: ImageMagick and GD

Images are converted on the server: the original you upload becomes the sizes your templates ask for (the *image aliases*). The book names two converters, ImageMagick (a separate program, the more powerful) and GD (a library inside PHP, a fallback). Both are still there, in `settings/image.ini`. The difference now: GD is first in the list (it works inside PHP and starts no program), and ImageMagick is the fallback for formats GD cannot read, such as TIFF or PDF, and for filters only it has. To prefer ImageMagick again, override the list in `settings/override/image.ini.append.php`:

INI

```
[ImageConverterSettings]
ImageConverters[]
ImageConverters[]=ImageMagick
ImageConverters[]=GD
```

The book's old note about GD before version 2 (poor scaling) is history: every GD in a PHP that runs Exponential is version 2 or later. The setup wizard turns ImageMagick off by itself when it finds no `convert` program.

Cron jobs

The book asks for a job every 15 minutes that runs `runcronjobs.php`, for workflows, notifications and other periodic work. Same today. The book's line passed `-C` to PHP; it is not needed, and the options that matter are `-q` (quiet) and `-s <siteaccess>`. The cron table of After installing, 10.3 is the current one, with a separate line for each part (`frequent`, the default, `infrequent`, `cache_cleanup`). Without the job the site works, but workflows that wait for the job and notifications by e-mail do not.

Configuration files

The book's explanation is still exactly right: settings live in `.ini` files; the defaults are in `settings/`; the folders `settings/siteaccess/<name>/` and `settings/override/` change them; the order is **override, then siteaccess, then the defaults** (the first value found for a setting wins, from the top). The files a builder touches most are as in the book: `site.ini`, `override.ini`, `image.ini`, and `i18n.ini` for languages. The file name should end in `.ini.append.php`. The reason the book gives is still the reason: if a web server ever served the file, it would run as PHP, and the settings are written as PHP comments, so nothing is shown.

Where it changed: you rarely need to edit those files by hand. `./console exp:ini` changes a setting with a backup and shows the difference (Changing settings with exp:ini), and the administration has a settings page (Settings in the admin). The details of the order are in Settings scopes and their order.

Next

The setup wizard, then and now.

Source and license

Adapted from chapter 1, "Installing eZ publish", of *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license). Changes: shortened and rearranged; every step, path and setting checked against Exponential 6 and updated; the product is named Exponential.

The setup wizard, then and now

The twelve pages of the Learning book's setup wizard, matched to the pages of the Exponential 6 wizard, with what changed (database choices, site packages, siteaccess methods) and the commands that skip the wizard.

The book walks through the wizard page by page. The wizard is still there, with more pages and more choices, and two commands can fill it in for you. This page lines up the book's pages with today's. The exact screens, fields and messages are in The setup wizard of the installation book; read that when you install.

> **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.

The pages

Book (3.x)	Exponential 6	What it asks
1 Welcome	Welcome	The language of the wizard. A **Finetune** button appears when an optional test failed.
2 System check	System check	Directories the server must write (`var/`, `settings/`, and more), PHP version and extensions, `memory_limit`.
(none)	System finetuning	Hints for the optional tests that did not pass.
3 E-mail settings	Outgoing Email	How mail leaves the site: sendmail, SMTP, or the file transport that only writes mail to files.
4 Choose a database	Choose database system	MySQL or MariaDB, PostgreSQL, SQLite, MongoDB, Oracle.
5 Database initialization	Database initialization	Host, user, password, database; what to do when it already holds data.
6 Language support	Language support	The default language and any others; add more later in the administration.
7 Site packages	Site package, Package language options	A ready-made site to install.
8 Site access configuration	Site access configuration	URL, port or hostname.
9 Site details	Site details, Site administrator	Title, URL, database name, administrator account.
10 Site security	Site security	The `.htaccess` file, when you do not use a virtual host.
11 Site registration	Registration	Optional.
12 Finished	Creating sites, Finished	The addresses of the new site and its administration.

What the book's advice still means

The system check. The book says to give the web server write access to `var`, `settings` and `design`, with `chmod 777` as the quick (and, on shared hosting, unsafe) way, and `chown` to the web server's user as the better way. Keep the better way: give the web server's user (or the PHP-FPM pool's user) ownership of the directories it must write, and nothing wider. Never `777` on a server with other users. See Permissions and ownership.

Choose a database. The book's advice "if you do not have a reason for PostgreSQL, you probably do not need it" generalizes: choose what you can run and back up. SQLite needs no server and is right for trying a site and for small ones; MySQL or MariaDB is the common production choice (Databases).

Languages. Unchanged: do not add languages you are not sure of; you can add one later.

Site packages. The book lists eight packages (Blog, Corporate, Forum, Gallery, Intranet, News, Plain, Webshop) and recommends *Plain* to build from scratch. The packages of an Exponential installation are different: the default is `sevenx_multisite`, and the packages the installer offers are those found in the package repositories it can reach (Packages). For a site built from nothing, choose the smallest package offered and read Building a site without code.

Site access. The three access types of the book are unchanged. **URL** (`/admin`) is the simplest; **port** and **hostname** need web server configuration (and DNS, for host names). Every installation now gets three siteaccesses: `site` for visitors, `admin` for the administration and `editor` for editors. See Siteaccess settings.

Existing data. When the database already holds data, the wizard asks what to do, as in the book: add to it, remove it, leave it. Back up before you choose.

Skipping the wizard

- `./console exp:install --apply` installs from the command line, with the database given as options (Console install).
- The kickstarter reads `kickstart.ini` and answers the wizard's pages for you (Kickstarter).

Next

Installation problems, then and now.

Source and license

Adapted from chapter 1, "Installing eZ publish", of *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license). Changes: the pages matched to Exponential 6; advice on permissions, databases and site packages brought up to date.

Installation problems, then and now

The two problems the Learning book's first chapter troubleshoots (PHP memory limit, PHP as CGI) and how they look in Exponential 6, with where to look first.

The book ends its first chapter with two common problems. Both still happen in some form.

> **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.

PHP memory limit

Then. The default PHP limit was 8 MB, too little. The symptom was a fatal "Allowed memory size ... exhausted" message. The book raised it to 16 MB with `php_value memory_limit "16M"` in an `.htaccess` file and said that 32 MB would be plenty.

Now. Exponential needs far more, and the setup wizard's system check looks at it: below 128 MB is a failure and below 256 MB a warning (System information). Set `memory_limit` where PHP reads its settings, which depends on how PHP runs:

PHP runs as	Where to set it
PHP-FPM or CLI	`php.ini` of that PHP, or a pool file; reload FPM afterwards
Apache module	`php.ini`, or `php_value` in `.htaccess` (only when the host allows it)
Exponential Velocity	the PHP that starts it; its own `php.ini`

`php -i | grep memory_limit` shows the command line value; the page **Setup > System information** shows the value the web server's PHP uses, which is the one that matters for the site (the two are often different). A command line script such as an import may need more than a page does; run it with `php -d memory_limit=1G bin/php/<script>.php`.

PHP running as CGI

Then. Whatever link you clicked, you got the front page. The cause was how old versions read the request path when PHP ran as CGI instead of an Apache module, and the fix was to edit one kernel file.

Now. This symptom is rare with PHP-FPM or an Apache module. When you see the front page for every address, look at the rewrite rules first: all requests that are not a real file must reach `index.php` (see the shipped `.htaccess`, and the row "404 for every page under Apache" in Troubleshooting). The page **Setup > System information** shows how the web server runs PHP (the SAPI), which is the way to tell a module from FPM or CGI.

Where to look first

1. `var/log/error.log` (Where the logs are).
2. The debug output for yourself, not for visitors.
3. Installation troubleshooting, the full list by symptom.

Back to the Learning book chapters.

Source and license

Adapted from chapter 1, "Installing eZ publish", of *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license). Changes: both problems restated for Exponential 6 with current limits and tools.

What is content management?

What a content management system does (collect, manage, publish), what that means in Exponential, and the design step that comes before any click: planning the structure.

Content management is organizing, categorizing and structuring information so that it can be stored, published and reused in several ways. A **content management system** (CMS) collects, manages and publishes content. The book describes four things a CMS does, and each one has a place in Exponential:

A CMS should	In Exponential
Store content as parts (with metadata) or as whole documents	Content is stored as **objects** made of typed **attributes**. A document, such as a PDF, is a file attribute of an object. See classes, attributes and objects.
Keep the links between parts	Objects relate to each other, and one object can sit in several places of the tree. See the node tree and sections.
Keep track of versions	Every object is versioned. See versions, translations and related content.
Run a workflow: approvals, notifications	Workflows run on events such as publishing, and notifications go out by e-mail. See workflows and triggers.

How the content gets out

The book names the protocols: HTTP for the browser, and WebDAV and SOAP for other programs. Today the site and administration use HTTP, and programs use the REST API (REST API). The text format is still a structured XML of headings, lists, tables, links and embedded objects (rich text tags).

Design before you build

The book's first practical advice has not aged: **plan the content before you build it**. There are no fixed rules, only good practices from the people who build taxonomies for document collections. A short list:

1. Write down the kinds of content (articles, products, people, events) and what each one needs. These become content classes (creating a content class).
2. Sketch the tree: what belongs under what. The top entries often become the main menu.
3. Decide who may do what: groups and roles (roles and policies, step by step).
4. Decide the sections: parts of the site that need a different look or different rights.
5. Only then create the first class.

For a small project with experience behind you, the book allows starting at once with a rough outline made in the administration. For a large project, take the time at the blackboard. The pages Designing your content model and planning an example site work through it.

Next: the node tree and sections.

Source and license

Adapted from chapter 2, "Content Management with eZ publish", of *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license).

Changes: rearranged and shortened; settings, labels, datatypes, tags and URLs checked against Exponential 6 and corrected; links to the current guides added.

The node tree and sections

Content objects versus the nodes that place them in the tree, how a branch inherits its section, and what sections are used for (permissions and presentation).

Objects and nodes

The structure of a site is a **tree**. The book's example tree holds a document branch, ordered galleries and a forum. The key idea of this chapter is a distinction that is easy to miss and important to keep:

- A **content object** is the information itself: its attributes, versions and translations. It has one object ID.
- A **node** is a *placement* of an object in the tree. It has a node ID, a parent node and a place in the order.

One object can have more than one node, that is, appear in several places, without being copied. Delete one placement and the object stays where it is still placed. The name belongs to the object, and is shared by the versions of one translation. In the administration, the places of an object are its **locations** (the **Locations** part of an item in **Content structure**), and one of them is the **main location**: the one whose address is the object's own.

The address of a node on the site comes from its names along the path (see Links, URL aliases and wildcards); `/content/view/full/<node id>` is the plain form and always works.

Sections

A **section** is a virtual collection of nodes that belong together by meaning or by function. Sections do two jobs:

1. **Permissions.** A role can be limited to a section ("editors may edit articles in the News section").
2. **Presentation.** A template can look at the section and show a different menu or even a different page layout for it.

Rules the book states and that still hold:

- A node belongs to one section.
- A new child node **inherits** the section of its parent, so assigning a section to the top of a branch sets it for what is created below it later. (What already exists below needs **Assign content** on the section's page, which gives the section to the chosen item and everything below it.)
- There is always at least one section, the default one.

How to create and assign sections, and what the section also controls, is in Sections and Sections, roles and access for builders.

Try it

1. Open **Content structure** and note the tree on the left.
2. Open any item and look at its **Locations**: the main location, and any other.
3. Add a second location for an item (the **Locations** window of its edit form) and see the same item appear

in two branches. Change the title in one place, and both show it.

Next: classes, attributes and objects.

Source and license

Adapted from chapter 2, "Content Management with eZ publish", of *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license). Changes: rearranged and shortened; settings, labels, datatypes, tags and URLs checked against Exponential 6 and corrected; links to the current guides added.

Classes, attributes and objects

How a content class defines a kind of content through its attributes, how objects are created from it, and how versions, translations and the version limit work.

Class, attribute, object

- A **content class** says how one kind of content is structured: an article, a folder, a person.
- A **class attribute** is one field of the class, of a given **datatype** (a short text, rich text, a date, an image). A datatype is the lowest level of typing, though several are powerful compound types (the datatypes tour).
- A **content object** is an instance of a class. Its content elements are the attributes the class names. Besides those it has generic properties: object ID, name, creation date, owner and more. The name is built from one or more attributes by the class's **object name pattern**.

Exponential ships a large set of datatypes, and extensions add more (tags, a map location, star ratings).

Versions

An object is always under **version control**. Editing makes a draft; publishing makes it the current version and the previous one an archived version. The history is kept within a limit, and can be changed in `content.ini`:

INI

```
[VersionManagement]
DefaultVersionHistoryLimit=45
VersionHistoryClass[1]=75
```

`DefaultVersionHistoryLimit` is the number of versions to keep for every class; `VersionHistoryClass[<class id>]` sets it for one class (here, class 1 keeps 75). Two things from the book still hold:

- A limit below **2** is not honored: editing needs a draft beside the published version, so the kernel keeps at least 2.
- Older versions are not edited; to go back, copy the older version to a new draft and publish it. For a site with quality rules for documents, this is the document history.

Leftover drafts (an editor who left a window) are cleaned up by settings in the same section (

`DraftsDuration` and related) and by a cronjob; see Drafts and pending items. The book noted that comparing versions needed a diff tool. Today the version view shows the versions side by side (Content history).

Version management applies to **every** attribute, binary files included.

Translations

An object can be translated into other languages. An attribute that should not be translated (a photograph, say) can be marked so at the class level. See Content languages and Translating content.

Where to go on

Content model and editing is the reference for this in the administration. Next: content and presentation.

Source and license

Adapted from chapter 2, "Content Management with eZ publish", of *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license). Changes: rearranged and shortened; settings, labels, datatypes, tags and URLs checked against Exponential 6 and corrected; links to the current guides added.

Content and presentation

Why content, logic and presentation are kept apart, and the two places a page is made: the page layout and the content view.

One of the basic properties of the system is the separation of **content**, the **application logic** that handles it and its **presentation** for a browser or for other output (a PDF, a feed, an API). The same object can be shown as a full page, a line in a list, a teaser, a feed entry, or JSON, each by a template for that purpose.

Two areas of a page

A page has two main parts:

- The **page layout** (`pagelayout.tpl`): the frame around everything. Menus, header, footer, scripts and styles live here. In the book's figure these are the parts marked P.
- The **content view**: the part for the thing asked for (an article, a folder listing, a form, an administration screen). Content views come from the kernel modules for the address you asked for. These are the parts marked C. The content view is put inside the page layout as `$module_result.content`.

Examples of content views: an article, a folder with its children, a search form, role editing, the output of an extension's module.

Structure is not layout, but they can agree

The tree is independent of the page layout. Yet a well-planned tree can also drive the layout: the top-level nodes can be the main menu, and their children (mostly folders) the entries of a sub-menu. Templates can read the same objects the same way everywhere. See menus.

Where to learn the templates

- Templates for builders
- View modes
- The design chain
- The template language

Next: roles and policies, step by step.

Source and license

Adapted from chapter 2, "Content Management with eZ publish", of *Learning eZ publish: Building Content Management Solutions* by Paul Borgermans, Tony Wood, Paul Forsyth, Martin Bauer, Björn Dieding and Ben Pirt (additional material by Bruce Morrison), second edition, 2024, Book-Learning-eZPublish. Copyright (c) 2004-2024 7x and Graham Brookins. Licensed under the GNU Free Documentation License, version 1.3 or any later version, with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts (the license). Changes: rearranged and shortened; settings, labels, datatypes, tags and URLs checked against Exponential 6 and corrected; links to the current guides added.

GNU Free Documentation License 1.3