

SCRAP (Simple Core for Rendering Assets & Pages) CMS

SCRAP is copyright © 2026 Paul Ledbury, paul@ledbury.co.uk

1. Overview

SCRAP is a lightweight, mobile-responsive, flat-file Content Management System, consisting of one short PHP file and a single stylesheet, with no Javascript, no dependencies, and no database.

SCRAP is styled to be simple. Basic HTML markup is used for content pages, with the overall site layout (header, logo, menu, and footer) added automatically by SCRAP. The basic configuration (site name, logo, site owner, and menu structure) may be changed by editing a short JSON configuration file. The look and feel may be further modified by changing the core HTML page or stylesheet, as you wish.

2. Features

- **Flat File CMS.** No database, so editing is possible with a text editor, and backups and migrations are as simple as copying files.
- **Simple Configuration.** Configuration uses small JSON files for easy editing and rapid deployment.
- **HTML Markup.** Minimal page markup is required, using familiar HTML tags.
- **Page Type Options.** SCRAP handles three distinct types of page layouts: Standalone Article Pages, Dropdown Menu Pages, and Dynamic (Blog-style) Index Pages.
- **Subfolder Architecture.** All website content is neatly centralised inside a core **pages** folder, with subfolder names matching URL paths, and subfolder structure matching the site structure.
- **Self-Contained Bundles.** Pages are modular. An individual page folder houses its layout text (**index.htm**), local media assets (images, PDFs, Zip Files, etc), and optional JSON configuration files all in one place.
- **Base URL Detection.** SCRAP automatically detects and adapts to its installation folder (web root or a subfolder), so it can be installed in a subfolder for testing or site development, and easily moved for immediate deployment.
- **Secure Routing.** URL parameters are dynamically parsed, with built-in security measures to strip directory traversal subversion attempts.
- **Page Not Found Fallback.** If a requested page does not exist, SCRAP gracefully falls back to a custom page-not-found layout.
- **Site Map Generation.** SCRAP will dynamically generate a site map page as well as create **sitemap.xml** and **robots.txt** files.

3. Presentation & Styling

SCRAP's visual theme is powered by a lightweight, single-file stylesheet designed with modern layout standards and fluid responsiveness, providing:

- **Mobile Responsiveness.** The navigation dynamically adapts to mobile screens, swapping standard navigation for a CSS-driven hamburger menu (three-line stack) expanding into a clean, drop-down menu container, without requiring heavy external JavaScript frameworks.
- **Asset Scalability.** Standard content containers apply global width overrides on embedded content graphics to guarantee layouts never break width boundaries on mobile touch screens.
- **Boolean Logic Representation.** HTML `<not>` tags are styled into a custom overbar dynamically positioned above text for Boolean **not** in technical or logic-focused copy.
- **Code Representation.** HTML `<code>` tags are styled into a monospaced bordered line of code, or if surrounded by `<pre>...</pre>`, a block of code.

4. License

SCRAP is free to use and is released under the terms of the [GNU Affero General Public License](#) (AGPL).

5. Installation and Usage

- **Installation.** Extract the contents of the zip archive either to the root of your web space or to a folder for testing. The archive contains the core SCRAP CMS files (`index.php` and `style.css`) along with a routing file (`.htaccess`), example configuration file and logo (`scrap.json` and `scrap.svg`) and an example web site (the contents of the `pages` folder).
- **Routing.** The `.htaccess` file contains the directives needed for Apache to re-write the page URLs to the files used by SCRAP. If Apache is configured to use `.htaccess` files, you can keep the file in your CMS installation folder alongside the PHP and CSS files. Otherwise, copy the directives from `.htaccess` into a `<Directory>` section for the installation folder, in your Apache configuration file (`httpd.conf`) or virtual-hosts configuration file (`httpd-vhosts.conf`).
- **Configuration.** Edit the configuration file `scrap.json` to set up the site name, logo, site owner, and menu structure.
- **Content.** The `pages` subfolder holds the site content, with a subfolder for each page. Each subfolder holds an `index.htm` file for the page text, and any additional asset files needed (images, PDFs, Zip files, meta-files, etc). The structure of the folders here reflects the structure of the site. A sample web site is included in the download, which demonstrates the various options supported by SCRAP. See the section on *Page Types and File Structures* for more information.
- **File Editing.** SCRAP does *not* contain a built-in file editor. If you want a self-hosted solution then [Tiny File Manager](#) is a very capable PHP-based file manager and editor that can be installed alongside SCRAP. A better and more secure solution for remote access is to connect to a remote SMB share over a [WireGuard](#) VPN, and edit your files using your editor of choice.

6. Basic Configuration

Basic configuration (site name, logo, owner name, and the menu structure) is contained in the `scrap.json` file. For example:

```
{
  "title": "SCRAP CMS",
  "logo": "scrap.svg",
  "owner": "Paul Ledbury ",
  "menu": ["Home", "Projects", "Services", "About"]
}
```

The menu array items may either be a textual menu option which links to a folder of the same name (as shown above), or a nested array defining a dropdown menu (see *Dropdown Menu Pages* for an example).

7. Page Types & File Structures

Each page is a subfolder of the main **pages** folder, with an `index.htm` file containing the text content for that page. Each subfolder page also contains any assets relevant to that page. Standard relative links may be used in the page text, like `` or `<a href="brochure.pdf">`. These links will automatically be intercepted and rewritten by the PHP engine to include the correct absolute path, ensuring they never break.

Subfolders are named the same as their corresponding menu item, in lower case, and with spaces replaced by hyphens. So an **About Me** menu item would be contained in an **about-me** subfolder of the main **pages** folder.

Page markup is HTML, but pages *only* need HTML for essential markup. No page header or footer is required, and no `<head>` or `<body>` sections are otherwise needed.

For example, the `index.htm` in the **about-me** folder could simply be:

```
<h1>About Me</h1>

<p>Semi-professional Stack Overflow copy-paster.</p>
<p>I write code, then stare at it until it works.</p>
```

SCRAP handles three distinct types of page layouts based on menu and folder organisation: Standalone Article Pages, Dropdown Menu Pages, and Dynamic Index Pages, as detailed below.

The example web-site contains examples of each of the different types of pages.

Type I: Standalone Article Pages

Single pages that are used for standard, single-focus content pages that appear at the top level of the menu (such as a "Home", "About" or "Contact" page). The layout file and any media are placed directly inside the page folder.

```

cms/
└─ pages/
   └─ about/
      ├── index.htm      <-- The page content/HTML
      ├── team-photo.jpg <-- Image asset used only on this page
      └── brochure.pdf   <-- PDF asset used only on this page

```

Type II: Dropdown Menu Pages

Drop-down menu pages are a simple hierarchy defined by the menu in the **scrap.json** configuration file. They are used where a dropdown menu lists and links to multiple sub-pages. The names of the containing folder and the subfolders are the same as the menu item names in the dropdown menu.

```

cms/
└─ pages/
   └─ services/
      ├── web-design/
         ├── index.htm      <-- The web-design page content/HTML
         └── webdesign.jpg  <-- Image asset used only on this page
      └── consultancy/
         ├── index.htm      <-- The consultancy page content/HTML
         ├── example1.jpg  <-- Image asset used only on this page
         └── example2.jpg  <-- Image asset used only on this page

```

A dropdown menu is defined by a nested array containing the dropdown name and the dropdown options, in the **scrap.json** configuration menu:

```

"menu": [ "Home", "Projects",
          ["Services ▼", ["Web Design", "Consulting"]], <-- Dropdown menu
          "About" ]

```

Type III: Dynamic Index Pages

Dynamic index pages are a more complex hierarchy, defined not by the menu structure, but by the subfolder structure and meta-data within the subfolders. They are used when a parent page needs to automatically list and link to multiple sub-pages, whilst displaying a summary of each sub-page (such as a **projects** landing page displaying an index of various distinct projects).

```

cms/
└─ pages/
   └─ projects/
      ├── order.json      <-- (Optional) Controls display sequence
      ├── project-one/
         ├── index.htm
         ├── meta.json    <-- Text data (Title, description, thumbnail)
         └── thumbnail.jpg <-- Landing page preview image
      └── project-two/
         ├── index.htm
         ├── meta.json
         └── thumbnail.jpg

```

Embed the PHP function `GeneratePageIndex()` in the page's HTML to output the page index. For example:

```
<h1>Projects</h1>
<?= GeneratePageIndex(); ?>
```

Dynamic Index Page Meta-Data & Ordering

When using a Dynamic Index Page, SCRAP builds the index listing for the page using the following configuration files:

The `meta.json` File

Meta data describing the project or page contents. Required in every subfolder of an index page to populate the index listing:

```
{
  "hidden": true|false,      <-- (Optional) Hides the page from the index
  "title": "Modern Eco-Home",
  "subtitle": "Sustainable Architecture",
  "image": "thumbnail.jpg",
  "description": "A look at the latest solar-powered housing layouts."
}
```

The `order.json` File

Placed in a parent folder to manually control the display sequence of sub-pages in a folder index. A list of folder names in a json array, in the desired order of appearance:

```
[
  "project-two",
  "project-one"
]
```

Any folders omitted from this list automatically drop to the bottom of the index list.

8. Pages Not Found

A custom “Error 404 – Page Not Found” page can be added by creating a page in a subfolder of the `pages` folder named `404`. The page can be created with local assets in exactly the same way as any other standalone article page.

The page currently requested can be referenced in the error page by using the PHP `PageName()` function. For example:

```
<h1>Page Not Found</h1>

<p>The page '<?= PageName(); ?>' could not be found.</p>
```

9. Site Map

SCRAP will automatically generate a site map of pages, viewable at the `/sitemap` or `/sitemap` URLs. The site map is dynamically generated when the page is viewed. At the same time, SCRAP also automatically creates the files `sitemap.xml` and `robots.txt` for the site.

10. Logging

SCRAP will keep a simple log of visitors in the file `scrap.log` in the installation folder. If you do not wish this to be viewable by visitors, you can block it by adding the following lines to your `.htaccess` file or `<Directory>` section of your Apache configuration file (`httpd.conf`) or virtual-hosts configuration file (`httpd-vhosts.conf`).

```
<Files "scrap.log">
    Require all denied
</Files>
```

11. Random Quotes

Pages can include a random quote pulled from a `quotes.json` file containing an array of different quotes. For example:

```
[
    "The <i>only</i> way to do great work is to love what you do.",
    "<b>Life</b> is what happens when you're busy making other plans.",
    "In the middle of difficulty lies opportunity."
]
```

Use the PHP `RandomQuote()` function to insert a quote into the page. The function allows HTML styling tags to be used in quotes, and the stylesheet automatically inverts italic tags for italicised quote integration:

```
<h1>Home</h1>
<?= RandomQuote(); ?>
```