AI.COM “Micro CMS”

Version 0.7.8.1 – February 20, 2019

Introduction

This is a very basic Content Management System that I made for my website, fdossena.com.

In its current state this project is no good for inexperienced users, but it’s not meant to replace Wordpress, it’s just a very small CMS for people who want a nice looking Portfolio/Blog website.

Features

- Based on modern technologies
- Very lightweight
- Responsive
- Indexable by search engines
- Compatible with browsers as old as IE6 with minimum limitations
- Printing support (for articles)
- Articles can be written in HTML or Github-flavored Markdown
- HTTP and HTTPS (a certificate is required)

Features that I am not planning to introduce in future releases

- Any form of user authentication
- Article editor

Features that may be introduced in future releases

- Administrative area to add/remove articles and comments

Installation

Requirements

- Apache Web Server with PHP and MySQL
- A few Mbytes of free space

How to install

- Extract the contents of *AI_MINI* to the root of your server (usually it’s the *htdocs* folder)
- Enter PHPMyAdmin or another MySQL administration software
- Create a new database, empty
- Import *db.sql* into it
- Add an user with SELECT, UPDATE and INSERT permissions.
- Edit *_config.php* and insert the parameters for accessing the database (username and password of the newly created user, database name and the DB server address)
- Save

Basic customization

At this point, the site will have a Home page, an “About me” page and 2 sections (Projects and Blog) with an article each. All the content is placeholder text so you’ll have to fill it.

Essential Info

- Edit *_config.php* and change Site_Title, Site_Description, Site_Keywords and Site_Author to match your site
- Save
- Replace *favicon.png* and *favicon.ico* with the icon you want for your site (both 128x128)
- Replace *campaign-icon.png* with the image you want to show when a link to your site is shared on a Social Network (~512x512)

Editing the Home page

- Edit *home.frag* and replace the placeholder text with whatever you want to show on the Home page
- Replace photo_home.jpg with a picture of you (~1080x1080, <400KB)
- Save

  Later we’ll see how to add links to sections to the home page.

Editing the “About me” page

- Edit *about.frag* and replace the placeholder text with something about you
- Replace *photo.jpg* and *photo_small.jpg* with a picture of you. The first one can be any resolution, but the second one should be smaller (~800x600, <100KB)
- Save

Editing the Privacy Policy

- Edit *privacy/index.frag* and insert your own privacy policy. You can see an example here: http://fdossena.com/privacy
- Save

Creating an Article

The easiest way to create a new article is to start by copying one of the 2 included examples, and change it to your needs.

- Copy one of the examples
- Edit *index.frag*. Keep in mind that ALL paths have to be relative to the root, because the .frag file is loaded by *index.php* (or *basic.php*), which is at the top level. An article can link to another .frag file, using a link like `<ai_link frag=”path/page.frag”>Text</ai_link>`. <ai_link> tags will be processed into <a> tags at runtime. Never link directly (a href) to a .frag file. Needless to say, .frag files can be .php files if you need to use PHP.
- Save
- Enter PHPMyAdmin or another MySQL administration software and add a new entry into the *articles* table. Parameters:

- *Section*: the id of the section you want to add the article to (by default, 1=Projects, 2=Blog)
- *Frag*: path to the .frag file of your article
- *Description*
- *Icon*: path to an icon file (~200x200) for the article list. Only shown when the site is in basic mode. If left null, noicon.png will be used
- *CampaignIcon*: image that will be shown when the article via social media (~512x512). If null, default campaign-icon is used
- *Cover*: path to an image (~1000x600) for the article list. Only shown when the site is in full mode. If left null, nocover.png will be used
- *Title*
- *Date*
- *Relevance*: a decimal between 0 and 1. Articles in a section are sorted by relevance first, then date.
- *Kwords*: keywords separated by commas (for search engines)
- *UpdateFreq*: how often the article is updated. If left empty, it will be set to never. Valid values are daily, weekly, monthly, yearly and never.
- *Views*: view counter. Do not touch
- *Featured*: date of last time the post was featured or null if was never indexed. Post with the most recent featured date is featured

Links with <ai_link>

All links should use the <ai_link> element, which is processed at runtime and will be automatically turned into regular links for indexability by search engines.

Internal links (to other .frag files) can use this syntax:

```
<ai_link frag=”myPage.frag” ..other attributes..>Text</ai_link>
```

External links can use this syntax or the regular a href:

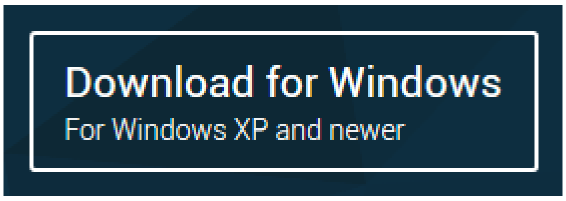
```
<ai_link ext=”http://example.com” ...other attributes...>Text</ai_link>
```

These elements are processed client side when the site is in full mode, or server side when in basic mode. Note that server side, the PHP DOMDocument parser is used, which may screw up your pages if they're not valid XHTML.

If you've used a previous version of this project, your links with loadFragment are still valid.

These links also work in Markdown articles.

Square download buttons



You can style an *a* or *ai_link* element to look like a typical square button with text inside by applying the *downloadButton* class. Here's an example:

<ai_link class="downloadButton" ext="http://example.com" ...other attributes...>

Main button text

<div class="small">Smaller button text</div>

</ai_link>

Note: you cannot put a link within a link. Some browsers allow it, others don't.

Adding/Removing Sections

Sections do not actually exist, *articles.php* simply fetches all articles from the DB that match a given section id, therefore, adding and removing sections is very simple.

- Edit *nav.php*: this is the file that contains the menu on top of the screen.
- You will see lines that look like this:

```
<ai_link frag="articles.php?s=2">Blog</ai_link>
```

- Simply add/remove such lines. Make sure each section has its unique id (shown in red here) and name (shown in blue here)
- Save

As a direct consequence of this mechanism, it is possible to implement "secret" sections by adding articles with a section id that's not in the menu and loading it manually using `loadFragment('articles.php?s=id')` in your browser's console (usually F12). Note that this provides no real security.

Additionally, you might want to add a link to the provided home.frag:

- Edit *home.frag*
- Inside the *sections* div, add an ai_link to the new section
- Save

Comments

This micro CMS has a built-in simple comments system with no login.

Each .frag file can have a comment section, by simply adding `<div id="article_comments"></div>` where you want to create the comment section. Note that that can only be 1 comment section per page. The provided examples already have this implemented.

Older versions of this project used the id _comments_ instead. This is still supported and will automatically be replaced with article_comments at run time.

The comments system is protected against SQL Injection and XSS attacks.

In the DB, comments are stored in the *comment* table. Each comment has a pointer to the page where it belongs to (the .frag file) and the comment text.

Removing comments

At the moment, comments can only be deleted from PHPMyAdmin or some other MySQL administration software.

Activating Disqus

Disqus is disabled by default. It can be activated from *_config.php* and it will replace the builtin comment system*:*

- Set *$Disqus_Enabled* to true
- Set $Disqus_Shortname to your site's shortname. You get the shortname when you register your site to Disqus

Note: comments cannot be transferred from the simple comment system to Disqus and viceversa.

Also note that Disqus is proprietary software and heavy. It's convenient because most users like it, but Richard Stallman would piss on your website!

Sharing links

You can create an area with share links and buttons by simply adding `<div id="_share_"></div>` where you want to create that area. Links for Facebook, Twitter and Google Plus will be created, as well as a text field with the URL.

Customizing the background

Edit *_config.php* and change *Background_JS* and *Background_ClassName* to select one of the backgrounds.

- Starfield (default) http://fdossena.com/?p=warpspeed/i.frag

  $Background_JS="BACKGROUNDS/warpspeed.js"

  $Background_ClassName="WarpSpeed"

  $Background_Config="{speedWhileLoading:18}"; //optional Note that this is a JS object, NOT A JSON STRING!

- Triangles http://fdossena.com/?p=muhTriangles.js/index.frag

  $Background_JS="BACKGROUND/muhTriangles.js"

  $Background_ClassName="MuhTriangles"

  $Background_Config="{saturation:0.8,customHue:202,lightness:0.25,outline:false,gradientType:"random",gradientIntensity:0.7,gradientMode:"smooth",gradientInvert:false,speed:1.5,instability:0.7,density:0.75,responsiveDensity:true,model:"mesh",fps:22}"; //optional. Note that this is a JS object, NOT A JSON STRING!

You may also want to change Chrome_TabColor to match your new style. This color is visible on Android Chrome.

If you want to create your own background, a template is included in the BACKGROUNDS directory.

Changing the footer

Edit footer.php and replace it with your code. PHP can be used freely but make sure it's compatible with both basic and full mode. Styling can be changed from main.css

More customization

The system uses a few .css files that you can edit to change the appearance of the website. These files are:

- *main.css*: contains the rules for full mode
- *basic.css*: contains the rules for basic mode
- *basic_overrides_ie.css*: contains fixes specific for IE<9 (for instance, using .eot fonts instead of regular .woff ones)
- *article.css*: rules common to articles
- *article_md.css*: rules common to markdown articles
- *thumbnail.css*: rules for thumbnails and gallery

Basic mode and Full mode

A site based on this project will have 2 modes: Full and Basic.

Full mode (index.php)

- For modern browsers
- Responsive
- AJAX
- CSS Animations
- Animated background
- Article list with covers
- Simple comments or Disqus
- Code highlighting

Basic mode (basic.php)

- For old browsers and search engines
- Not responsive on very old browsers, partially responsive on relatively newer browsers
- Static
- No animations on very old browsers, some animations on relatively newer browsers
- Static background
- Simple article list

- Disqus may not work properly (officially requires IE10+)
- No code highlighting

Compatibility

On fdossena.com, Basic mode is compatible with very old browsers (IE6+), while full mode requires a 2013-2014 browser at least (IE11+).

Needless to say, YOU are responsible for writing .frag files that keep compatibility with older browsers.

Links are interchangeable: if a user with an old browser receives a link that points to the full version, it will be redirected to the basic version, and vice versa.

Search engines and bots will only see basic mode.

Javascript must be enabled -even in basic mode- to ensure full functionality. With JS turned off, only very basic functionality is provided.

If you use Disqus, it officially supports only IE10+, therefore it may not work properly on older browsers and there's nothing that can be done to fix it.

Writing articles with Markdown

Github-flavored Markdown can be used for writing articles. More details about the syntax can be found here: https://guides.github.com/features/mastering-markdown/

Files that end in .md (loaded through a link, ai_link, loadFragment, …) will be treated as Markdown.

Some processing is done to your .md file to better integrate into the Micro CMS:

- Images (first level only) are added a lightbox automatically
- ai_link elements are parsed
- code blocks are applied formatting and highlighting
- links (both the ones created by markdown and the "a href" ones) will be set target="_blank"

Important: paths in links and images are relative from the root of the site, not to the .md file!

Creating .frag files from scratch

Basics

The first thing to do is to create a new empty file, and give it a name, such as *test.frag*.

The basic structure of a .frag file is very simple:

```
<div>
	CODE
</div>
```

Inside that div, you will put all your content. Content is organized in stripes. Each stripe element contains a content element, where you will put your text, images, ...

```
<div>
```

```
	<div class=”stripe”>
		<div class=”content”>
			Actual content
		</div>
	</div>
</div>
```

There can be more than one stripe, of course, but there should only be 1 content element per stripe. Use stripes to separate large sections of your page (like the article from the comments).

A handy template

```
<div>
<link rel="stylesheet" type="text/css" href="article.css" /> <!--optional, adds CSS rules useful for creating articles-->
<style type="text/css" scoped="scoped">
	/*optional: put your custom style rules here*/
</style>
<div class="stripe">
	<div class="content">
		<h2>Article title</h2>
		<p>Paragraph text</p>
		<img src="articleDirectory/img1.jpg" class="clickable blockImg" onclick="openLightbox('articleDirectory/img1.large.jpg')" />
		<p>More text</p>
		<h3>A subsection</h3>
		<p>A paragraph</p>
		...
	</div>
</div>
<script type="text/javascript">
	//optional: put your custom js here
</script>
<div class="stripe">
	<div class="content">
		<h2>Share this article</h2>
		<div id="_share_"></div>
	</div>
</div>
<div class="stripe">
	<div class="content">
		<h2>Comments</h2>
		<div id="article_comments"></div>
	</div>
</div>
</div>
```

Javascript functions that can be called from .frag files

isBasicMode()

Returns true if in basic mode, false otherwise. This is the easiest way to find out whether you are in basic mode or not using JS.

isDesktop() / isMobile()

Returns whether the user is in dekstop or mobile view. This does not necessarily guarantee that the user is using a certain type of device, it just says which view is being used, it could be a small window on a desktop, or a big tablet in desktop mode.

hideNav()

Hides the menu on top. Will be shown again when a new .frag file is loaded.

showNav()

Shows the menu on top if it was hidden.

hidePage()

Hides the .frag file. Will be shown again when a new .frag file is loaded

showPage()

Shows the .frag file if it was hidden.

openLightbox(url)

Opens the image at the specified URL in a lightbox. In basic mode, it opens it in a new window.

closeLightbox()

Closes a previously opened lightbox. In basic mode, it does nothing.

flash(color)

Flashes a certain color. Color can be any CSS color, expressed as a string.

loadFragment(url, pushState)

Loads a .frag file at the specified URL. The second parameter should always be true or not defined. If set to false, the new .frag file won't be added to history and it won't be possible to go back/forth between pages (useful for transitions within subpages of the same article).

In full mode, the .frag file is loaded via AJAX, but in basic mode it will trigger a full page reload.

setBackgroundCfg(cfg)
Sets a new background config, stores it in localStorage and applies it by reloading the background page. In basic mode, it does nothing.

onFragUnload()
This method can be overwritten by .frag files to perform an action when the frament is unloaded (for instance, remove some function added to the window object.

It can be overwritten in this way: window.onFragUnload=function(){ ...code… }

Keep in mind that if this throws an exception, it will crash the loading of the new frament.

I(id)
A short way to write document.getElementById(id).

loadText(target,url,onDone,noEscape)
Loads a text file of any type via AJAX, escapes special characters, puts the result into the target element, then calls the onDone function. If noEscape is set to true, the loaded text is not escaped (useful for loading a short piece of HTML)

highlight(target,language,preformatted)
Applies highlight.js to the target element, with the specified language.

All languages supported by highlight.js can be used, and they are dynamically loaded when needed. Available languages are all the ones that are listed under HLJS/langs. Notable exceptions: for HTML, use XML, for JSP, use Java, for C/C++ use cpp.

Note that this method will replace the target element's CSS classes.

In basic mode, highlight.js is not applied, only basic styling.

If applying to a <pre> element, set preformatted to true.

Local CSS
A .frag file can define its CSS rules that will only apply to that fragment. To do so, simply add this at the beginning of your .frag file, right after opening the initial `<div>`

```
<style type=”text/css” scoped=”scoped”>
        CSS RULES
</style>
```

Importing a CSS file
To import a .css file, simply add this at the beginning of your .frag file, right after opening the initial `<div>`

```
<link rel=”stylesheet” type=”text/css” href=”path/to/file.css” />
```

Local JS

Running a script when the .frag is created
Simply add this code somewhere in your .frag file (inside the `<div>`, of course)

```
<script type=”text/javascript”>
        CODE
```

```
</script>
```

Defining JS functions to use in the current .frag file

Simply add this code at the beginning of your .frag file, right after opening the initial `<div>`

```
<script type=”text/javascript”>
        window.functionName=function(param1, param2, …){
                //FUNCTION CODE
        }
</script>
```

Importing a JS file

Due to browser limitations, `<script type=”text/javascript src=”path/file.js”></script>` works slightly differently than you would expect, because the scripts are running while the JS you're importing is still loading, so you need to wait for it to be loaded before you run your code. There are 2 solutions:

- Waiting for a specific class to be loaded. This is especially useful for libraries. Example:

  ```
  var interval=setInterval(function(){if(!!window.MyClass){
      clearInterval(interval);
      //code that depends on MyClass
  }.bind(this),100};
  ```

- Using XHR to load a JS file, then using eval(xhr.responseText). See this: http://fdossena.com/?p=html5cool/loadAsync/i.frag

“Am I in Basic or Full mode?”

JS

isBasicMode() returns true if in basic mode, false otherwise

CSS

When in basic mode, the fragment is inside an element that has class `basic`. The code below will only apply when in basic mode.

```
.basic myRule{
        …
}
```

PHP

Simply check if the URL contains basic.php.

```
if(strpos($_SERVER['PHP_SELF'],’basic.php’)!==false){
        //basic code
}else{
        //full code
```

```
}
```

basic_only and basic_hide classes

Some elements in your .frag files can be shown/hidden in a specific mode.

```
<div class=”basic_hide”>
        <div class=”stripe”>
                <div class=”content”>
                        <h2>An exclusive stripe</h2>
                        This stripe is only visible in full mode.
                </div>
        </div>
</div>


<div class=”basic_only”>
        <div class=”stripe”>
                <div class=”content”>
                        <h2>Filthy old browsers</h2>
                        This stripe is only visible if your browser A SHIT!
                </div>
        </div>
</div>
```

Testing a page in basic mode

If you’re using a modern browser, the site will switch to full mode automatically. To prevent this, open the console (F12) and type: `localStorage.noSwitch=true` and reload the page. When you’re done, type `delete(localStorage.noSwitch)` and reload to go back to full mode.

Alternatively, if you’re on Windows 7 or newer, you can download this portable Internet Explorer 6: http://downloads.fdossena.com/geth.php?r=ie6-en (Will probably trigger your antivirus, if you have one). This won’t work in Wine. If you’re on an other OS, use a Virtual Machine with Windows XP.

Embedding a full webpage

Iframes can be used to embed other pages. Due to really crappy support from older browsers, it is best to use them only in full mode.

Using iframe.frag

This code can be used in a script or in a link.

```
loadFragment(‘iframe.frag?p=url’)
```

Using an iframe

```
<iframe src=”url” style=”width:100%”></iframe>
```

Licenses

This project is licensed under the GNU GPLv3 license.

Frameworks

Nothing

Libraries

Uses Highlight.js, under the BSD License.

Uses Parsedown, under the MIT License.

Uses WarpSpeed.js under the GNU LGPL License (by me)

Fonts

Uses the Roboto font by Christian Robertson (Google)

Uses the Bebas Neue font by Fontfabric

Uses the Lato font by Łukasz Dziedzic

Additional notes

The pictures of Garrus used as placeholder are unlicensed and were found online on DeviantArt. Garrus Vakarian is property of Bioware Corp.