AI.COM “Micro CMS”

Version 0.4.1 – May 2, 2016

Introduction

This is a very basic Content Management System that I made for my website, AdolfIntel.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 aimed at people like me who want a 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

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
- Save

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

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 `<a onclick="loadFragment('path/page.frag')">Text</a>`. Never "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. If left null, nullicon.png will be used
    - *CampaignIcon*: image that will be shown when the article via social media (~512x512). If null, default campaign-icon is 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.

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:
  `<a onClick="loadFragment('articles.php?s=2')">Blog</a>`
- 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.

Comments

This micro CMS has basic support for user comments that doesn't require login.

Each .frag file can have a comment section, by simply adding `<div id="_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.

Keep in mind that comments are not shown when the site is in basic mode.

The comments system is protected against basic SQL Injection (uses parametric queries) and XSS attacks (uses HTML Purifier).

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.

Customizing the background

This project uses MuhTriangles.js as the background.

Customizing the background

- Edit *_config.php* and change *Background_DefaultConfig.* If you want to see how to customize MuhTriangles.js, see the project here: http://adolfintel.com/index.php?p=muhTriangles.js/index.frag
- Save

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 all the basic rules to display the documents. It is included in both basic and full mode.
- *basic_overrides.css*: contains changes to main.css that only apply to basic mode (for instance, using .eot fonts instead of regular .woff and .otf ones)
- *articleList.css*: style for the *articles.php* page, to show the list of articles in a section
- *article.css*: rules common to articles
- *thumbnail.css*: rules for thumbnails

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
- Comments

Basic mode (basic.php)

- For old browsers and search engines
- Not responsive
- Static
- No animations
- Static background
- No comments

Compatibility

On AdolfIntel.com, Basic mode is compatible with browsers as old as IE6, while full mode requires 2012-2013 browser at least (IE10, FF12, Chrome 30).

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.

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”>
			Text…….
		</div>
	</div>
</div>
```

There can be more than one stripe, of course, but there should only be 1 content element per stripe.
Titles can be added inside the content element using the h2, h3, …, h6 elements.
The content is organized in sections, each with 1+ paragraphs.

```
<h2>My page</h2>
<div class=”section”>
	<p>First paragraph</p>
	<p>More text…</p>
</div>
<img src=”path/img1.png” />
<div class=”section”>
	<p>Some text</p>
</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.

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)

Loads a text file of any type via AJAX, escapes special characters, puts the result into the target element, then calls the onDone function.

highlight(target,language)

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.

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

Unfortunately, due to browser limitations, it is not possible to use `<script type=”text/javascript src=”path/file.js”></script>` like you would on a normal page. This is because the scripts are running while the one you're importing is still loading. There's a workaround for that, and it involves using a timer (with setInterval) to check periodically if the script is loaded (for instance, when a certain method becomes available) and then run the code that requires that script. I did this for my speedtest: http://speedtest.adolfintel.com

Alternatively, just use an iframe to embed a full webpage with a transparent background, the user won't notice.

“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(strops($_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>
```

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 HTML Purifier, under the GNU LGPL v2 license.

Fonts

Uses the Roboto font by Christian Robertson (Google)

Uses the Forte Line font by Paul Kirtzmire

Uses the Bebas Neue font by Fontfabric