Mamur a mythological beetle like creature which shares our philosophy; useful, small, very hard working and servers it's master faithfully.

Mamur Content Server V1.05

Introduction

Mamur makes building and maintaining web sites faster and cheaper. It is very easy to learn and it is possible to start creating professional web within minutes. Unlike a lot of systems it is not just easy to start with but time consuming to produce professional results; Mamur is a fairly, small and simple system and focuses on doing only the key tasks. It is not over bloated with functions which may never be used but it is not too trivial; as you can see from the tag definition tables. There are many tags which you do not need to be bothered with most of the time but on occasions they can be vital to the completion of a project. Then plug ins allow the system to be upgraded in increments or customised for specific tasks. The core system is therefore kept lean and mean and generally applicable.

Mamur can best be thought of as, a content level template system with an integrated content server, a basic Content Management if used with a CMS plug-in or as a development framework. The template system it is not indented to provide all advanced application requirements and sometimes integrating in Smarty or a Zend Framework should be considered. This can be achieved by using Mamur plug-ins.

Mamur on its own will serve web sites but the editing and creation of content has to be done using traditional HTML editors and tools such as photoshop. However, just install a plug-in and you have a full CMS when your require it. An important point to note is that the CMS plug-in- is loosely coupled it can simply be disabled or deleted without affecting the running of live web site. It would be easy to build a site with Mamur CMS and then ship it to another server so that it can either to be remotely updated or run without a CMS.

The Problem with Hand coding Pages without Mamur

When producing web sites a common problem is that most pages share a common layout and branding and if each page duplicates these shared areas it becomes much harder to build and maintain a web site. It is quite common that a simple change might have to be patched into every page on the web site or if design templates are used the entire web site will have to be updated offline and re-loaded back to the web site. This can also lead to additional down time, upload issues, support and development costs.

Mamur Content Server offers a compromise for users which are happy to build pages and templates using html editors such as Dreamweaver or perhaps just a text editors. It is easy to install and above all an easy to learn; you can to get to grips with it within an hour. It runs as a very fast, small software foot print solution, which does not need and maintenance or frequent upgrading. Graphics

Designers do not need to see any php tags or code and can work independently from software engineers.

Reducing the risks of using a CMS.

1 Reliability

When basic content is stored only in a database a database server failure will take down the entire site. Mamur pages are more reliable because they can only fail when the web server fails. No matter how highly reliable a database server is, the reliability of the web server and database server together will always be less. A Mamur CMS system can still run when the database is down because the database is not required for serving standard pages. For many sites this will mean no failure is visible to the public and for others which might require a database for advanced features the web site can implement an alternative solution or degrade gracefully.

2 Manual updates and corrections

By being able to get quick access to page content using traditional methods it is easy to correct errors made by a CMS editor or to patch in a feature which is not yet supported but required urgently.

3 Safer/cheaper Back-ups

The Mamur pages are automatically backed up by routine server backups and are less likely to fail on recovery since they use standard files formats and a corruption in one file only affects that item of content. Databases can easily be corrupted if backed up whilst being updated, not backed up entirely or there is a corruption in one of the files. Data recovery from databases can be costly especially if the web site is down. Also verifying each backup will restore is an additional expense.

How it works

Mamur Content Server is contained within a single folder which is placed in the root directory of your web site. When activated a .htaccess is installed for you which causes all request for html files to be either obtained by the Mamur server or direct from your web directory as before. If the Mamur server knows about the html page requested it will generate it for you using a template and content files. If you can't start from scratch, is possible to start with a standard web site and upgrade page by page even while the site is live.

Content and Page Files

Mamur knows about a page when in the /mamur/user/pages folder there is placed a folder with the same name of the page followed by _html. Inside this directory there must be valid page.xml and one or more .html files which contain content for the page. For example a url request for /index.html causes the mamur server to look in /mamur/user/pages /index_html/ for the page details. If found it servers the page using a defined shared template and the page contents stored inside the page folder. If not found Mamur will see if there is a standard web page by that name and server that instead or if nothing can be found it will report a page not found error using a custom template. Sub-folders can also be used for example /buy/product.html would map to /mamur/user/pages/buy/product_html/.

Not all content is page related and for content which can be shared such as blogs, articles, sections, menus and structures (blocks of html formatting) are stored in separate content folders in the shared area. It is even possible to share files between sections of the web site differently so that for example news, menus and styles will be automatically selected according to the area of the web site visited.

Applications and dynamic content can also be easily created and inserted into a page just like other

pieces of shared content. Advanced applications can also plug-in to the system to extend features or features to multiple pages automatically.

Templates

Templates can be found in mamur/user/templates. The default one is called main_template.html and it is often possible to build an entire web site by installing your standard web page in this file. All you then need to do is place special place-holder markers to indicate where you would like page titles, meta tags, page content, shared content and the output from php scripts. These place-holders are called tags and you will see below that they are fairly simple pieces of text. You will only need to know a few to get started but there are quite a number of them available in order to do some quite clever stuff should the need arise.

Page Title, SEO and Templates (page.xml)

The page.xml file inside each Page configuration folder defines the page title, meta tags, any other important SEO details, special pages scripts and styles. It can also defines a template so you can use as many templates as you want and location for shared content files. For people not use to XML the file this need not be a concern as you can use an existing file as an example and make simple text changes as you would with html files.

Tag replacement no assembly required.

Some systems build pages in pieces which is hard to debug, test and modify. Our approach is to make a skeleton page with minimum content and insert into it style sheets, java, php and content. This means that the overall page template and the inserted content are complete in themselves; making building and support much easier. This approach masks php code from designers and php engineers can get the code working and even styled while the overall page design is being perfected.

Templates and any included html content and php files can call in various pieces of text, html, php etc by using a simple place holder tag which starts with '[?:' and ends with ':?]'.

The following tags are recognised:

format is [?:tag attribute=”value” :?] - some tags allow no attributes some are mandatory and some tags may have many attributes e.g. [?:tag attribute=”value” attribute2=”value” attribute3=”value” :?]

attributes with the same name cannot be repeated within a tag,

The attributes (e.g. file=string) can be unquoted, double or single quoted i.e. file= string or file=”string” or file='string'. Unquoted strings cannot contain spaces and no string may contain “:?]” sequence but may contain these characters separated by other characters.

Tags which insert just a string of text characters:

Tag	Attributes	Description
cookie	name	The inserts the contents of a named cookie which is escaped for security ie htmlspecialcharacters escaped:

		$_COOKIE [name]
get	name	The inserts the contents of a named get (ie url variable) which is escaped for security ie htmlspecialcharacters escaped: $_GET [name]
global	name	A global is defined in the configuration file. The value to be inserted in place of this tags is defined in the configuation xml file. Plug ins may simply do this in the background so they just appear as strings known to the web site by name. They should not be used by code or plug-ins to store lots of information and should be limited to a few important strings such as store name, address, contact email or important configuration such as non-delivery days.
home	*(no attributes are required)* ref *(optional)*	Mainly used to help create links and urls or perhaps in text where you need to print the web site name or a web page reference. Prints the home page url **without** a trailing '/' eg https://mywebiste.com use: [?:home :?]/ for https://mywebsite.com/ can set parameter "ref" to add a location to make a full url (provided you do not need spaces) eg [home ref="/dir/mypage.html?x=1" :?] prints https://mywebiste.com/dir/mypage.html?x=1 if leading '/' is ommited one is added for you eg [?:home ref=index.html :?] is alo a home page on a linux system
mamur	*(no attributes are required)*	Insert the mamur relative location ie /mamur is the default location for a standard set up. Used to allow mapping via a url to the mamur accessible folders and files. Using [mamur :?] instead of /mamur allows the default configuration to be changed without breaking any links.
nonce	*(no attributes are required)*	Number used Once is a security id. Add this to gets, posts, ajax interfaces and cookies which need to be protected against re-use or an attack where even if a logged is required a user may be tricked into causing an unintended act. When this tag is printed a unique 16 digit letter and number sequence is returned. The number is saved in current session and will be deleted when the session expires. Scripts and built in functions and plug-ins can get the nonce with a call to $this->mamur->getNonce(); or $this->mamur->getNonce('name') - typically a delete or update which can be submitted by a user should have a nonce and this should be matched by the actioning script to ensure that

		it is valid. The script may call setNonce to ensure that the action cannot be repeated or if repetition is not an issue can rely on session expiry to limit potential attacks. Note: When using Ajax the page may not refresh the nonce so the back end call should not reset the once on every call or should pass back a new nonce. The Ajax call will fail the nonce if the session expires although the page may still be live so the Ajax return should also handle this.
	length *(optional)* name *(optional)*	Attribute length can be used to alter length of nonce (default=16) Attribute name can be used to distinguish different concurrent nonce values used by different applications. If not used the name 'default' is assumed and you can use functions getNonce and setNonce with no parameters.
page_selected	name *(optional)* dir *(optional)* url_part *(optional)*	If the page name matches name attribute the tag prints "_selected" otherwise nothing is printed. Can be used in menu code to change style of selected page eg [?:page_selected name=pagename :?] [?:page_selected dir=admin :?] In a multi-level directory system menus may need to indicate which sub-directory has been selected. The dir attribute matches the directory part of the url. If the url has deeper levels then the match will still be made provided the levels given match eg [?:page_selected dir=admin/web :?] would match http://x.com/admin/web/anypage.html and http://x.com/admin/web/deeper1/deeper2/anypage.html In the lowest level sub-menu always use page and full dir to ensure that the correct page is indicated. url_part attribute matches from after the domain name starting with or without a leading '/' for the length of the string given up to and including any the file extension. May be used to achieve matches on parts of directory and filename. Also, can be used with name and dir attributes to mask out undesired matches
odd_even	*(no attributes are required)* name *(optional)*	This tag will alternately print "odd" or "even". Used with styles can strip columns and rows of tables. An optional attribute "name" allows different sequences on same page for example one might be called row and the

	set *(optional)*	other column. If not set “default” is assumed for the name. An attribute called “set=” allows the sequence to restart from either odd or even or if set to some other value will use that value first then followed by “even”. A new page restarts the sequence.
post	name	The inserts the contents of a named post (ie form variable set using post method) which is escaped for security ie htmlspecialcharacters escaped: $_POST [name]
request	name	The inserts the contents of a named variable (ie a form variable set using post or get methods, a url variable or cookie) which is escaped for security ie htmlspecialcharacters escaped: $_REQUEST [name] note: request makes it easy to receive variables by different methods and is easier to forge so has some security implications.
title	*(no attributes are required)*	Inserts the title text as defined in the page.xml file. The tag can be used any where but typically would be used between title tags in a template eg <title>[title :?]</title>

Tags which insert content from a file:

Tag	Attributes	Description
article	*(must use one of the following attributes)*	This is for content which is page independent and can be used as a article in any appropriate page. It should be as a HTML 5 article would be. An article should, in HTML 5 start and end with a article tag which should not be used for styling and should start with a title. Located in shared/article folder ie Has as a file reference relative to: user/shared/article
	name	If name is used it will refer to a file in the directory with extension .html.
	file	The file attribute allows a full relative file location to be given to a sub-directory.
	dir	Looks in directory for a file matching the page name.html

<table>
<tr><td></td><td>mapped</td><td>or uses default.html

The mapped attribute allows content mapping based on url directory or subdirectory so that it is possible to map different url directories to different content files such that each directory shares a the same content file. The simplest is one level deep and is used to have a different content file based on the first level directory given in the url:

eg mapped=”mapdir/nodir” means :

File Location in shared content — url

.../mapdir/nodir.html — if the url has no directory

.../mapdir/first_url_dir.html — if the url has one or more directories the the file will be named as the first level directory

.../mapdir/unknown.html — if the url has one or more directories but content cannot be found

It is possible to map to any directory level ie If mapped= ”mapdir/firstlevel/nodir” then the mapping will have a first level directory as a folder in mapdir and within that directory the filename will map to the url second level directory etc.</td></tr>
<tr><td>blog</td><td>(must use one of the following)

name

file

dir
mapped</td><td>This is for content which is page independent and can be used as a blog article in any appropriate page. It should be as a HTML 5 article would be but allows blog type articles to be separated for some reason. May be one type is sent to a feed differently
A blog should, in HTML 5 start and end with a article tag which should not be used for styling and should start with a title. Located in shared/blog folder ie
Has as a file reference relative to: user/shared/blog

see article for explanation of mapped attribute.</td></tr>
<tr><td>menu</td><td>(must use one of the following)

name

file</td><td>This is used for menu content to be placed in the page. Menus are often shared between templates and there can be more than one menu on a page or content file.

All content is in shared/menu folder ie
Content with menu html as a file reference relative to: user/shared/menu</td></tr>
</table>

	mapped	see article for explanation of mapped attribute.
section	*(must use one of the following)* name file mapped	This is for content which is page independent and can be used as a section in any appropriate page. It should be as a HTML 5 section would be. Section content should in HTML 5 start and end with a section tag which should not be used for styling and should start with a title. Located in shared/ section folder ie Has as a file reference relative to: user/shared/section see article for explanation of mapped attribute.
structure	*(must use one of the following)* name file dir mapped	This is used for shared structure components such as parts of page headers, footers, content columns , Miscellaneous block of styles , JavaScript ,in the body of a page. Most commonly used to simplify templates so that the common parts are saves as separate structures. **This content is not Editable** with standard content editor. Located in shared/ structure folder ie Has as a file reference relative to: user/shared/structure see article for explanation of mapped attribute.
page_content	*(must use one of the following)* name file	Page content is specific to a page and is not content shared by other pages. Either a name or file attribute must be given and maps to the file in the default directory which is always the page directory ie for page /foo.html → user/pages/foo_html/ ie for page /bar/foo.html → user/pages/bar/foo_html/ Example content insert tags are: [?:page_content name=myname1 :?] is located in user/pages/bar/foo_html/myname1.html [?:page_content name=myname2 :?] is located iuser/pages/bar/foo_html/myname2.html You are free to use any name for content which makes sense eg main, right panel, column1 etc

Tags which have dynamic results such as page redirection:

Tag	Attributes	Description
http_header	*(none not allowed must use one of the following)*	Headers such as cache and redirect sent before the html page is returned as part of the messaging protocol used for HTML. Headers must be sent before the page but buffering allows this tag to be placed anywhere in the document. However at the very top of the template or content is recommended.
	value	The tag must have value= eg value=”HTTP/1.1 404 Not Found” and a name for the header name.
	name	Alternatively instead of value the tag may have a name=”nameid” in which case it uses any definition found in the page.xml or template.xml files.
php	*(must have either name or file attribute)*	Has as a file reference relative to: user/php eg [?:php file='helloworld.php' :?] or [?:php name=helloworld :?]
	file *(option)*	file defines relative path within php content folder and file name and extension

	name*(option)*	name just defines the file name without extension and is a simpler tag for web sites which do not use multiple directories within the php folder
		UNLIKE content files you cannot place tags inside php files but you can process tags using php strings and function calls as illustrated by this example:
		$row=
		“<tr class='myrow [?:odd_even :?]'><td>$myRowData<td></tr>”;
		$this->processTags($row);
		$this->doPhpAndPrint($row);
		
		The first line sets up a row string with a mamur tag inside, note by using double quotes or a heredoc string any php variables are inserted at this stage.
		Then:
		$this->processTags($row); // function replaces mamur tags with php inserts
		$this->doPhpAndPrint($row); //function processes the php tag functions and prints.

	(optional) any user defined parameter other than file or name eg myvar=mayvalue	[?:php name='helloworld' mayvar=”hello” :?] $parameters=$this->model->getParameters(); returns an array of parameters defined in the associated php tag. This can then be used to programmatic change the output from the php code. NOTE:

		Advanced programmers should note that php tags are included into a view class; the php files contents are read and processed by a view function doPhpAndPrint. Within the php file $this refers to the view so you can use either $this->model to refer to mamur functions, however, if you may prefer to use $this->mamur to refer specifically to the mamur model MVC. Software engineers who are MVC pattern purest should use the php inserted code only for formatting output and should write a plugin to do controller / model like functions. Plugins are run within your own class outside of mamur. However, this is not enforced so for simple web applications it would be best to keep thing simple by placing all the code within one or two php files. Note: php can also be run from .php files placed in any accessible directory
protected	*(combination of one or more of the following)* allow_group disallow_group allow_status disallow_status allow_status_and_above allow_status_name disallow_status_name	Protects a page and causes redirection to a login page. The tag can be used in templates and if used in content the entire page will also be protected. The protected tag can also verify that the user has the required status or is a member of the correct group by setting the following attributes: allow_group, disallow_group, allow_status, disallow_status, allow_status_and_above,allow_status_name,disallow_status_name Also the login page is defined as login.html unless it is defined in configuration.xml to be another page. Optionally the protected tag can have an attribute “login_page=” to define a special log in page. The system does not provide any login page logic (see advanced section) but manges user login status according to 3 configurtation.xml settings: allowSessionCookie="yes" - allows login status to be saved loginTimeOut="900" - sets login time out period in seconds loginPage="login.html" - allows setting of default login page (note login_page= attribute overrides)

Tags which special effects:

Tag	Attributes	Description
other_js_files	*(no attributes are required)*	
other_css_files	*(no attributes are required)*	
other_meta	*(no attributes are required)*	

Data name index table row

tag name index

all type=page or shared content type, separator, match="*.html" - for listing contents

shared type= allows creation of new types of shared content.

option name= - for setting page remembered options eg Ajax to mdify a form

application	Application is reserved for a future use.
element	Element is reserved for a future use.
widget	Widget is reserved for future use.
page_data	reserved for system use to store information about a page. This tag is private and can only be set in a pages xml file – there is no associated tag available
page_timer	If turned on in configuration Prints " page Time {$pagetime} ms ". If placed at bottom of page shows the time spent by php in producing the page. Only use tag: [?:pagetimer :?]
page_timerms	As above but not conditional on configuration and just the numeric value in ms is printed. Only use tag: [?:pagetimerms :?]
top_dir_selected	If the top sub directory name matches the tag name then the tag prints "_selected" otherwise nothing is printed. Can be used in menu code to change style according to page directory. Only use tag such as [?:top_dir_selected name=directoryname :?]

page_url	prints full uri of the page (a url is the same as a uri in this context)
page_name	prints name of page without any extension
page_name_ext	prints name of page with any extension
page_ext	print page extension only
page_dir	prints just folder or directory reference for current page (no end '/')
date	prints time now in RFC 2822 format when = date time string eg “now” “next week” etc format = date format string as php date function
unique_serial	inserts a unique serial number
random	inserts a random string of numbers and letters default length =6 attributes = length , upper_only=upper_only

ADVANCED USERS ONLY

Tag mapping using XML files (advanced users)

For completeness we will mention here that all tags can be defined via page.xml however, when beginning you should skip the following more detailed explanation and use a simple scheme as mentioned above.

When a named tag is printed the page.xml file for the current page is searched for a matching element i.e. a matching tag name and name attribute. If there is no match the tag's attributes are used.

If a match is found the XML element may define the value to insert using a "value" attribute in a closed XML tags or by using CDATA or text between open tags. Eg <meta name="foo" value="bar" /> or <mata name="foo">bar</meta>

Alternatively, the XML element can have a "file" attribute to refer to a file location relative to the top directory for that element eg for the "body" (ie shared content in the body of the document) it is realtive to the folder user/shared/body.

When there is no matching definition in page.xml the name will be used to find the file in the top level of relevant folder using an assumed extension. This is most often used for page_content since mapping using xml file definitions generally does not add any clarity since the content is closely related to the page. However, in other cases using mapping in an xml file makes sense in that it keeps the tags short and easier to maintain if you need to move a shared_content file into a sub folder.

As mentioned page_content tags almost never require any mapping within the template or page xml.

Log in page (advanced users)

PHP sessions are not used for login status and user details. Instead an encrypted session cookie in set in the clients browser.

You will have to use a wordpress plugin to help manage login from wordpress or write your own login page or download a mamur plug-in.

From a plugin or php tag ie [?:php :?] , there are some system calls which help manage logins:

$this->mamur->userLogIn($status,$statusName) - will login in a user with an optional status

$this->mamur->userLogOut($status) - will log out a user setting an optional status

$this->mamur->setUser($name,$id,$loggedin,$status,$group,$statusName) - will set user $loggedin set to true if logged in. Status Name should be consistent with numneric status value

$this->mamur->getUser() – returns an array of the users details including login status

Persistent Page State using Data sets (advanced users)

Php sessions are not used, so you are free to write code to use sessions as you please. General purpose Plugins not written by a user should also avoid using php sessions. Instead the built in dataset method which is also available for users.

The dataset method works independently of php sessions and only introduce an overhead if used. It has the advantage of storing data only in named data sets. Providing a sensible naming convention is used, it is much less likely to cause a data conflict than the use of php sessions which encourage the use of lots of individual variables. Also plug-ins can hook into the clean up process and can permit some datasets to be retained for a longer period eg a shopping cart plug-in might use a shopping_cart data set, hook into the clean up process and extend the life of a shopping cart or

perhaps save it in a recovery folder.

A advanced application could hook in APC or Memcache to improve dataset access speed.

There is a built in system for manging data sets by sending and retrieving a multi-demensional data array using the following functions:

You must use get the data set array, set it after changes and delete it when no longer required ie:

$this->mamur->getDataSet($name) - gets a data array from the named data set.

$this->mamur->setDataSet($name,$data) - stores in a data set the data in $data array.

$this->mamur->deleteDataSet($name) - deletes a data set and removes it from session store

Alternatively a data set array supporting forms can be worked on directly using:

$this->mamur->setDataSetField($fieldName,$value,$dataSetName='',$tableName='',$record=0)

$this->mamur->getDataSetField($fieldName,$dataSetName='',$tableName='',$record=0)

Note the last used dataset name and table name will be used if omitted. If never set the name 'default' will be assumed.

When these functions are used a session persisting file is stored in the user database area.

In support of this feature please see the following configuarion.xml settings:

```
<!-- set to yes to allow session cookie to be set supports user login, admin login
     session id for use by plugins to save session datasets -->
<set allowSessionCookie="yes"/>
<!-- set to yes to allow permanent location id cookie to be set supports metrics
     allows returning visitors to be accessed when log plugin is used -->
<set allowPermCookie="yes"/>
<!-- set sessionTimeOut="3600" to clear down dataset sessions after 1 hr of inactivity
     this will cause lost of all dataset information unless a plugin traps the
     clean up attempt and extends the session, This has nothing to do with php sessions -->
<set  sessionTimeOut="3600" />
```

Plugins and Hooks (advanced users)

This is a fairly complex subject and best left for detailed discussion in a separate document. A plugin is a special class which is loaded if configured in the configuration.xml plugin section and the class installed in the plugins section. Technically, a plugin is a controller class which interacts with the model and view to alter the system behaviour, add additional tags etc. There are some examples which may help get you started.

Plugins may set globals and do other system calls eg

$this->mamur->setGlobal('name','massage');

Also plug-ins can hook into the system by registering a call back function eg

$this->mamur->registerSessionClearFunction($this,”my_function_name”); *

$this->mamur->registerPagePrintFunction($this,”my_function_name”); *

$this->mamur->registerUrlFunction($this,”my_function_name”);

$this->mamur->registerTagFunction($this,”my_function_name”);

$this->mamur->registerNamedTagFunction($tag,$this,”my_pre-process_tag_function);*

$this->mamur->registerSessionLogoutFunction($this,”my_function_name”);*

$this->mamur->registerServerBaseFunction($this,”my_function_name”);

* CAUTION these functions must return true and only return false under special circumstances.