qmd-CMS

A simple but versatile CMS

keywords:

CMS, PHP, MySQL (or other), OOP

harry graner AKA ***hy
Apartado 77
35508 Costa Teguise
Las Palmas Spain
www.qimanfaya.net
qmd@qimanfaya.net

Version: 0.8.6.30
Datum: 2014-05-06
Autor: ***hy
Status: draft
Doc: /media/work/www/qmdCMS/manual_qmdcms.odt

content

1 abstract

This document addresses the following audience: administrators setting up a cms.

This manual is not only intended for experienced web developers – but if you want to modify the code you should:

- understand the readme.html-file and other accompanying files.
- You should be able to maintain a mySQL-database

If you want to make changes to the system you should also be 'fluent' in

- HTML
- CSS
- PHP
- JavaScript (basic knowledge)

There is also a libre-office-spreadsheet called „qmd_notes“ with notes on some of the details. Please have a look at this one as well. This document also includes a 'Roadmap' for future development.

Frequent Abbreviations:

(F) = function ; E.g. (F) showItem() = function showItem()

1.1 disclaimer

In the current version (<1.0) qMD-CMS is not a fully working system. You should not install this to live web servers. The demo Database may contain nonsense–articles, that are provided in order to test specific functionality – or in order to demonstrate some use-cases.

Sorry for the poor structure of this document. I'll try to improve it, but will focus on developing 1.0 first.

This software is submitted as is. I.e without any warranty or promise regarding functionality. You are using this software at your own risk. No compensation for damage or loss of data or other … whatsoever.

There might as well be some flaws and issues. E.g. by deleting categories or menuitems the system will probably produce „orphaned articles“. So you better have a database-editor at your fingertips when rearranging and customizing the demo-data according to your needs …

1.1.1 design

I am NOT a designer – and the results are pretty poor ATM. So, you'll probably want to change css-styles. Please create your own schemes and contribute these schemes to qmd-CMS.

1.1.2 Security

I am an experienced developer and very security aware. But I am pretty new to building websites with PHP and I guess there are a lot of implications out there in the web. So my focus is to get this up and running and find some enthusiastic developers that help me improving it. I would be very grateful if someone would participate and improve matters in regard of security.

Again: You should not install this to live web servers – at least do NOT implement

- blogs
- forums
- multiuser
- … even comments may be an entry-point for crackers)-;

1.1.3 documentation

In addition to this file there is a

- readme.html-file, that also addresses installation and covers a few general questions
- A LibreOffice-Spreadsheet containing several notes on details, structures and DB-Tables as well as a roadmap for future development [Version 0.9, 1.0 …]

At the Moment I do not maintain seperate documentations for Users who want to just work with qmd-CMS, but there is one chapter adressing usage in this document.

Again: the program is currently **NOT** intended for **live-websites**. So the target audience are programmers that are interested in finding a simple but versatile CMS and improving/contributing to this software.

Sorry, if you don't find what you are looking for in this documentation – I am focussing on the code at the moment. If you have any questions please contact me: qmd@qimanfaya.net

… and if you don't find a solution to your problem: **Use the Source, Luke!**

Or have a look at http://www.qimanfaya.net/qmd-CMS

I'll try to make a commentable FAQ-Section and maybe a forum in the near future (i.e. late 2014)

1.1.4 Source

Many (F) require a lot of parameters – some of them default to specific values and you can omit them. Most of these functions have a comment in their headers where the parameters are explained. The more data you provide when calling these functions the faster they are – instead of collecting the required stuff via SQL-Statements …)-;

If you want to contribute to this please drop me a note. There is a Mercurial Repository on bitbucket, and you'll receive an invitation soon.

1.2 License

This code was developed by ***hy

1.3 Thanks

qMD-CMS is originally based on sNews [www.snewscms.com] which was developed by http://www.solucija.com

It was released under a creative commons license 2.5 [http://creativecommons.org/licenses/by/2.5]

I was using sNews 1.7.1 beta to customize it to my needs – more credits are given in the code.

I want to thank the people who developed snews and especially keyrocks + Rui Mendez who - amongst others - helped me in their forum (http://snewscms.com). Thanks to user „tucuta“ who provided the first version of a spanish language file.

1.4 Download the latest version

Before attempting to install qmd-CMS make sure you have obtained the latest version. ATM this product is under heavy construction so bugfixes may be entered on a weekly or even daily basis.

The default download site is bitbucket.

If there is no other recommendation all repository entries are not intended for productive systems.

Download a copy here:

https://bitbucket.org/hy_soft/qmd-cms/downloads

If the download section is empty (yes, it is empty at the moment) select the tab labelled „Tags“.

Do not use any of the repository prior to 'tip'. They may be broken, missing some files or other inconsistency ...

Get the repository:

Contact me for access to the repository. You'll be invited to bitbucket.

Create your local repository in a folder:

```
mkdir qmd
cd qmd
hg init
```

edit your file `[qmd/].hg/hgrc`

```
[paths]
default = https://hy_soft@bitbucket.org/hy_soft/qmd-cms
```

and then type:

```
hg pull && hg update default
```

Enter your password...

1.5 Install

Please follow the very few and simple instructions in the readme.html - file accompanying your copy of qmd-CMS.

1.6 Developers, developers!!!

If you want to participate you are welcome. Please drop me an eMail and I'll invite you to the repository at bitbucket.

What is needed most?

- Designs – better schemes, VAC-containers, a left hand side menu and more
- Javascript – cleaning up and improving the Javascript code which is a mess
- Security – check
- a nice way of displaying images to each article (such as carousel or similar)
- files/upload, search, sitemap (fails completely)
- testing

Please have a look at the roadmap in qmd_notes.ods or roadmap.ods

2 concept

In order to maintain content in an easy manner, most of the system is kept simple. My intention for this CMS was to have an easy system. Just enter articles and hook them to a menu. So from an admin point of view you layout the menu-structure and add the according articles.

Actually things got a bit more complicated, because an article can have more than just a title and a text. Furthermore the CMS should be SEO-aware (at least).

2.1 Menu-Driven

There is a **main menu**. The articles are just added to a menuitem and displayed when the menu in question is selected by the user.

You may add **submenus** and as many articles as you like.

If there are multiple articles for one menu-item they all are shown – as a teaser. The **teaser** is the 'Title' and the first part of an article which is separated from the rest of the article by a [break].

You can alter this behaviour.

2.2 Categories

Another aproach to divide the content (and provide a different access to articles) are **categories**. You don't have to use them, but it's giving you more flexibility to ease access to some featured articles.

2.3 Schemes

A scheme contains an index.php that will reflect the general layout of your site and css-Files defining the styles and colors. You can have as many schemes in your site as you want. Any category or menu can have its own schemes. An article is always assigned to a distinct scheme and any article is loaded within the context of the scheme that is assigned to the article.

2.4 Articles

Everything with content is an article. No matter if it is an advertisement (VAC) or a „true article“.

Once you have designed the layout of YOURSITE.COM, you only have to enter articles. Which is as easy as it can be.

Either you enter just text via copy/paste – or you do the same with some PHP/HTML-code.

You can also just include a valid php file:

```
[include]/content/myNewArticle.php[/include]
```

concept

2.4.1 Article-Objects

If you write your own article objects there will be more flexible ways to maintain your site.

[See also the section 5.3 Article-Classes on Page 16]

Additionally your articles may reside in another Database than the rest of qmd-CMS – if you make your own objects. All you need to do is to make a descendant that initializes the required database just by entering the name in the constructor:

```
class excursions_list extends qmd_base_article{

      function __construct($controller,$BaseRec){

            parent::__construct($controller,
                  $BaseRec,'mytablename','mydatabase');

      }

}
```

CAUTION: all comments will reside in the main database of qmd-CMS unless you overwrite the displayArticleComment-Function … which is not a big deal but the maintenance will be...

2.4.2 VAC – Value added container

A **V**alue **A**dded **C**ontainer may contain headers and links to articles or just display an advertisement e.g. an image. The content within a VAC is made up of articles that are tagged as VAC.

So if you want to display an advertisement just add the ad as an article and select VAC in the combobox on the head of article. VACs don't have comments enabled by default.

2.4.2.1 Issues

ATM the VAC will not really be handled as an article-Object so don't instantiate them.

2.5 Languages

ATM there are three language files included for English, German and Spanish.

For Administrators: a lot of (Error-)Messages are hardcoded in English and not to be translated. I expect admins to be able to cope with the english language.

2.5.1 Any Language

Sometimes your article may be available in any language. Instead of changing the language and inserting an article per copy and paste you may just have one version of the article in order to keep it easily maintained.

Just use the systemwide variable called: „$qmd_Lang_ANY“ which defaults to: „@ANY“.

The former is used in the code and the latter is used in the Database.

An article that is available in this language will not be translated. This tag is intended for banners or images (ads) that may not need any translation. It is easier to set them to lang='@ANY' instead of maintaining the same image/content in several different languages.

2.5.2 Add another language

The following section only applies to introducing a complete new language to qmd-CMS! If you want to translate an article into another language see: 6.3.6 Article-Languages on page 23

In order to add a language you have to do some stuff manually:

- create a new language file and translate from one of the already existing (./www/lang/*)
- add a row to table qmd_categories_ML with the appropriate name of the language
- add a row to table qmd_menu_ML -"-
- add a flag/banner representing the country of the language to ./www/dev.img
- make sure the language is within the language array in init_languages() in qmd_lib.php
- enter the name of your flag into the case-statement of (F) show_SelectLanguage

I hope this is all... I haven't tested it – yet.

2.6 differences to snews

This chapter is dedicated to users of snews-cms from which I've derived qmd-CMS.

Well, ATM it's impossible to describe and explain all the differences in detail. So, if you're not familiar to snews please skip this chapter.

You are still reading this – aren't you? You're welcome. I introduced some fancy debugging techniques [see 7.1 Debug] that might be of help.

Furthermore I introduced a function-wrapper called qmd_die(). It does the same thing as the normal die-command of php. But it also sends a system message to the admin.

These system messages are – more or less – available on a user to user basis... but not fully implemented.

There is no Extra-stuff and there are only articles no pages (as in snews). The term 'pagemode' in the context of this documentation refers to either:

- a php file that is included into an article.
- an article that is shown in the general context (header, footer, background) but without VACs [**=pagemode**]
- an article that is shown in an otherwise empty context – usually an included php-file [**=fullscreen**]

Additionaly we have **VAC**-CONTAINERS. These are displayed like ads on the header,left, right or bottom of the page. If you have a container on the right hand side of your web-site you can inlude all ads that are to be displayed on the right side by:

```
fill_vac_right();
```

2.6.1 Unique names

Contrary to snews-cms you can have the same name (seftitle) for categories as articles or categories and menu-entries(seftilte). Because when a category is called the URL will contain "cat/' and so the distinction can be made whether to call the category or the article/menu respectively. I found it quite exhausting to find different names for the same stuff over and over.

2.6.2 Design

There are schemes you can customize and use in order to present pages in different layouts/styles depending on topic/category or for each article...

concept

2.6.3 technical

2.6.3.1 OOP

There is some usage of OOP implemented. Some things are just easier to do with Objects... Articles for example.

2.6.3.2 Database

Actually a prefix is still suppurted as in snews. But the default prefix is “qmd_” and you can add another prefix. You are also able to include tables from a completely different database.

This will be described later in detail. If you need immediate help, please don't hesitate contacting me.

Furthermore I stripped all function calls marked as deprecated, like mysql_*

2.6.4 user interface

The administration is much simpler (in my eyes). As an admin you can directly edit articles and comments. After any operation there is a linklist with useful shortcuts to items you might probably need next (back to edit the same article, or view the edited article in the website and the like...)

2.6.5 W3C-validated

The pages that are produced are validated against the w3c validation.

Check your outputs at: **`http://validator.w3.org/check#`**

But anyway it might depend on the input you are entering. A simple menu-entry as “Food & Beverage” validates to an error because of the ampersand (&). You have to label it: “Food & Beverage”

2.6.6 Users & Roles

Multi-user is implemented. Any user is assigned to a role (Root, admin, Moderator, 2nd-level-User and regular User). ATM there is no difference between Root and admin. Moderator is similar to admin but can not create articles. He/she can only edit articles. 2nd Level Users can do the same as a regular user. The benefits are comments that are reserved to 2nd Level-Users – depending on your settings.

ATM you have to set user roles using your database-administration-tool.

One user is created automagically: system.

The user “system” has no privileges. This user is only used to send messages to the admin.

2.6.7 Comments

Comments can have subcomments. Quoting is enabled. You can report an inappropriate comment to a moderator. Comments are stored with the current language and you can chose to see only those comments in your language or all.

2.6.8 System-Messages

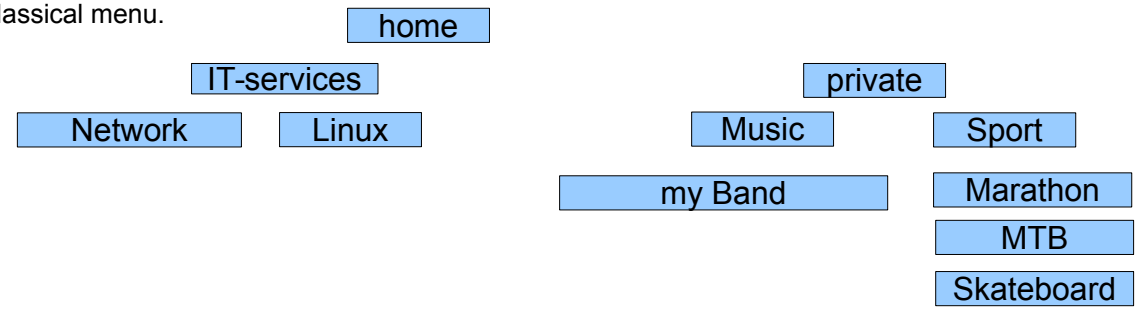
There are system messages sent from a user named 'system' to the admin. Instead of die() there is an alternate command: qmd_die() that will send the dying message to the admin.

You will recieve a notification about those messages after login.

3 Elements of qmd-CMS

3.1 Menu

A menu should be selfexplanatory. But the advantage of qmd-CMS is that you do not have to use a classical menu.



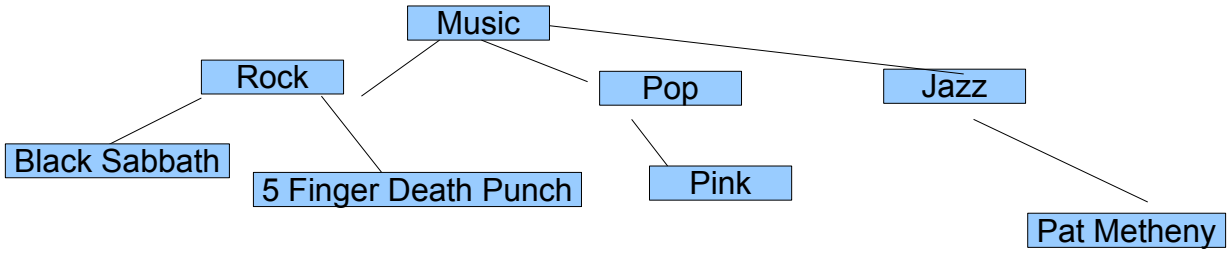
You don't have to use a lot of main-menu-items. You may have two metrostyle 'tiles' for two different sections of your website – e.g. IT-services and private. Only these two options are displayed on your homepage - according to the scheme of your main-page.

And each of your websites' sections may have its own layout (=scheme) and a different kind of menu. So the 'private-section' above would only show the part of the menu tree that descends from private.

echo build_menu ('private');

3.2 categories

Categories are intended to be maintained in a treestructure. You don't have to use them, but it's giving you more flexibility to ease access to some featured articles. Categories can be whatever you like.



3.3 comments

Comments may be made available to each article. You can turn them off at any time. You can display old comments even if you have disabled the comment function for an article.

3.3.1 general settings for comments

Your best bet is to have comments enabled in settings. So you can enable easily commenting for individual articles. The general settings only has three options:

- off
- frozen
- commentable

“Off” means no commenting enabled and no comments are shown – even if if there are any.
“Frozen” means **no commenting** allowed but old comments are shown – if if there are any.
'Commentable” means that comments are shown (if there are any) and commenting is enabled.

If you have commenting enabled and encounter SPAM or offensive comments you can just turn off commenting with one click – by changing the settings.

For programers: the following variables are to be set when you change these settings:

- table qmd_settings: commentMode (SITE-WIDE)
  2 (default) enabled comments
  1 frozen comments (show comments – don't allow posting of new comments)
  0 comments completely disabled

Only if the above variable is set to 2 (default) the settings of articles are applied:
The settings page has an additional option for default commenting on articles:
Enable Comments (default setting for new articles)

The combobox according to these default settings offers all options that an article can have (see below). This will be the default comment-state for new articles.

3.3.2 comments in articles

The table below show all options for commenting on articles:

		USERMODE		
#	explanation	no user	USER	2nd -level-User
0	not commentable/don't show comments	-	-	-
1	frozen – don't add new comments	see comments	see comments	see comments
2	commentable	see/write	see/write	see/write
5	users can see frozen comments	no comments	see comments	see comments
6	only users can comment	see comments	see/write	see/write
7	FROZEN but Users can comment	see comments	see/write	see/write
9	only 2nd Level Users see frozen comments	no comments	no comments	see comments
10	only 2nd Level Users can comment	no comments	no comments	see/write
11	FROZEN but 2nd Level Users can comment	see comments	see comments	see/write

3.4 schemes

Schemes are not to be misinterpreted in a manner that there are layouts offered on a user basis – as an opposite to a theme that a user can select. A scheme is just the main layout to display the site. Any article may use it's own scheme. E.g. in order to have a different background than the rest.

In order to make sense a scheme offers

- css-files where you can override the default qmd.css
- special JavaScript files
- images (usually background images...)

Schemes should be used if one or more article(s) share the same layout.

In order to get the best and quick results copy the (folder with the) default-scheme to default.org.

Try editing the default-scheme, change some css..., alter the index file … create new designs – and share.

A scheme requires at least an index.php-file.

3.4.1 Styles in schemes

If there is a css-folder in your scheme-folder and if it contains a css file named exactly like the scheme it is loaded automagically. E.g.: your scheme is called: **`/schemes/great_web_site`**
the css-file **`/schemes/great_web_site/css/great_web_site.css`** would be loaded automagically. You can change this behaviour in settings.

3.4.2 Javascript in schemes

If there is a js-folder in your scheme-folder and if it contains a javascrip-file named exactly like the scheme it is loaded automagically. E.g.: your scheme is called: **`/schemes/great_web_site`**
the css-file **`/schemes/great_web_site/js/great_web_site.js`** would be loaded automagically. You can change this behaviour in settings.

3.4.3 General notes on loading css-styles and JS

There are several ways to include JavaScript and css-files:

You can hard-code the required *.js and *.css into the article-object that requires (one of) these.

You can load automagically all (or just one) *.js and *.css-files that exist in
`/schemes/<myscheme>/js/*`
and
`/schemes/<myscheme>/css/*` by using settings

Last but not least the used /schemes/<myscheme>/-Folder is scanned for files named
load_css.php

and

load_js.php

If they exist they are being loaded. Please don't put anything else into these files but[1]:

```
global $qmdC;

$qmdC->load_js('my-js-file.js',TRUE,0,TRUE);
```

The css- and JavaScript-files are loaded to a queue and will be inserted into the HTML-page generated at the apropriate time.

[1] the second line that actually loads the file can be repeated with different names

3.5 Lists

[Also refer to next chapter]. A regular article consists (mainly) just of a title and a body text that should contain all required content – as images...

But probably you have data that needs more special fields to highlight basic information. If this kind of data is required for several items you should use lists.

E.g.: Excursions (see data in the demo-version)

Tour	**Type**	**km**	**Difficulty**	**Terrain**	**Technique**	**endurance**	**Time**	**Price**
Zonzamas	MTB	32	2	sand	2	3	3 h	28
Tinamala	MTB	21	1	gravel	2	3	2h 30'	25
Tahiche	MTB	25	1	gravel	2	3	3h 30'	30
Teseguite	MTB	29	3.	rocks	3	4	3h	35
Famara	HIKE	9		picon	2	2	4h	35
Caldera Blanca	HIKE	10	2	lava	3	3	5h	40

As you can see, the excursions feature a set of data that they all share. Instead of 'coding' this into every article and maintaining layouts of hundred articles it makes sense to create an object that will handle this data (for displaying, editing and more).

Furthermore the usual article-table in the database doesn't provide these fields, so you might have this kind of data in another table or even a completely different database.

All this can be handled with lists which is a derived article-object.

The demo data shipped with qmd-CMS contains a use-case with excursions.

4 The demo data included

Some notes on the articles that are provided with qmd-CMS

Some articles are shipped with qmd-CMS in order to give you a hint on the general usage.

4.1 Articles

4.1.1 readme

The readme file is also available as an article. It is a demonstration how to create a very simple and basic article-object on your own. In this case the only special thing that this article does is loading a specific css-file for this article.

4.2 VACs

There are some ads included in order to demonstrate how to handle them. You can delete all ads with your database-administration-tool:

```
delete from qmd_articles_ML where parent in (select id from qmd_articles where VAC>0);
delete from qmd_articles where where VAC>0;
```

4.3 Lists

There are Lists as well included in a demo-database ('excursiones'). It' just in case you want to know more about the look and feel of the possibilities.

4.4 customized contact-form

If you need a customized contact-form or other kind of interaction with the user (=FORM) you should have a look at:

`/qMDCMS/qmd_article_class.php` (the last few lines of code is a form-object)

`/www/schemes/excursions/exc_contact.php` an actual object (an overriden form)

5 Objects

Welcome dear reader to this section that is targeted to the developer.

I started this project off as a fork to snews-cms. So there is still lot of procedureal code in it. But I had at least some reason for implementing the main structure in OOP.

5.1 qmd_CONTROL

There is a global instance of qmd_CONTROL called „qmdC“.

This CONTROL-Object owns one or more Article-Objects.

In the initialization process the URL is evaluated and then article-objects are created and initialized.

The CONTROL-Object will read the properties of the article(s) in question and write them into the meta-data part of the HTML-header to be delivered to the browser. So search engines will be able to find the content...

The initialization process is described in qmd_notes_INIT.pdf which should accompany your version of qmd-CMS.

5.2 qmd_SQL

There is a global instance of qmd_SQL called „qSQL“.

This SQL-Object is used to do the database queries.

There is also a file named: qmd_db that has the global $DATABASE-Object.Which is where the connect to the database happens. Furthermore there are some functions in order to select, update, insert and delete stuff...

These functions are deprecated. I'll remove them... as soon as I get rid of them...

[Annotation: I wrote the above sentence some 7 months ago. I don't think I'll make it.]

5.3 Article-Classes

5.3.1 qmd_custom_article

This is an abstract base class that is never instatiated.

This Object implements only basic functions.

5.3.2 qmd_base_article

This is the actual default class for articles.

This Object is used as a base class for articles.

5.3.3 qmd_LIST

This is a descendant of *qmd_base_article* that is able to display LISTS.

This Object is able to display lists of similar items and display list entries as an article.

To work with your own customized lists you need to do the following:

5.3.3.1 Create tables

Create a table that will contain the common list data – e.g. Runningcompetitions:

Table „excursions“:

id	Tour	Type	km	Difficulty	Terrain	Technique	endurance	Time	Price
1	Zonzamas	MTB	32	2	sand	2	3	3 h	28
2	Tinamala	MTB	21	1	gravel	2	3	2h 30'	25
3	Tahiche	MTB	25	1	gravel	2	3	3h 30'	30
4	Teseguite	MTB	29	3.	rocks	3	4	3h	35
5	Famara	HIKE	9		picon	2	2	4h	35
6	Caldera Blanca	HIKE	10	2	lava	3	3	5h	40

A unique id-Field is required! The column labelled „Tour“ is not required, because we can use the article-title from regular articles. You can add more columns – such as „imagename“ that may contain (language independent) images for the item.

You do not need to create another table that contains stuff that needs some translations, because the body text is handled as in a regular article.

Every item in the excursions-table is loaded from a descendant of qmd_lists

5.3.3.2 Create an object/descendant

Put the code for your object in an otherwise empty php-file – e.g. „marathon-list.php“.

```
class excursions_list extends qmd_LIST{
      function __construct($controller,$BaseRec){
      parent::__construct($controller,$BaseRec,'competitions','');
      //now we also need to set some variables:
      $this->sql_list='SELECT x.*,a.id as artid,a.seftitle,ML.title
            FROM `excursiones` x
            left join ('._PRE.'qmd_articles a)on(a.id=x.parent_article)
            left join ('._PRE.'qmd_articles_ML ML)on(ML.parent=a.id)
            WHERE (1=1) '.$sWhere;
                  $this->URLfieldName='SEF';
                  $this->LinkFieldName='Name';
                  $this->IDField='id';
      }
}
```

URLFieldName: contains the fieldname that is used to generate a URL for the Article.

LinkFieldName: is the (Field-)Name that will have the URL underneath – so you can select the Item by clicking on the Name -Field in the list.

The **IDField** is required to find the proper comments for this item – if comments are enabled.

Furthermore you should overwrite the

```
function showListItem(){
      if (is_null($this->_selectedItem)):
                  $result=FALSE;
      else:
                  $result=TRUE;
                  //DO YOUR DISPLAY s7uff here!!!
      endif;
      return $result;
}
```

5.3.3.3 Instantiate your object

Create an article as admin. Fill the **title** of the article e.g. 'competitions' and **seftitle** 'competitions-list'.

Then fill the preloader with the path and filename of your php-file that contains your object (e.g. content/marathon-list.php).

Insert three lines in **qmd_classes.php**:

seek the (F) load_Article_Data()

Insert a line with your SEF into the case statement [**`switch ($loader)`**]:

```
case 'competitions-list':
      $result= new marathon_list($this,$r);
      break;
```

6 Administration

6.1 menues

6.1.1 deleting menues

If you are about to delete a menu that has submenus you are prompted to confirm deleting the menu or to delete all child menus as well.

CAUTION: If you select to only delete the menu – it's submenues won't be accessible through administration. You need to delete the dead menues by using your good mySQL-administration panel.

Such a dead menue may produce errors when creating another menu with the same SEF-title. It is also not possible to create a category with that SEFtitle.

So if you don't want them to be deleted, please hook them to another parent menu **before** deleting the menu in question.

You should prefer to chose deleting all submenus.

CAUTION: there may be orphaned articles as well after deleting menues. I probably need to fix this.

6.2 categories

6.2.1 deleting categories

If you are about to delete a category that has sub-categories you are prompted to confirm deleting the category or to delete all child categories as well.

CAUTION: If you select to only delete the category – it's sub-categories won't be accessible through administration. You need to delete the dead categories by using your good mySQL-administration panel.

Such a dead category may produce errors when creating another category with the same SEF-title. It is also not possible to create a menu with that SEFtitle.

So if you don't want them to be deleted, please hook them to another parent category **before** deleting the category in question.

You should prefer to chose deleting all sub-categories.

CAUTION: there may be orphaned articles as well after deleting categories – that are not visible to administration. I need to fix this.

6.3 Editing articles

There are lot of options when editing articles – but actually an article just consists of

- a title
- a seftitle/permalink
- and the text itself.

6.3.1 Three ways of setting up an article.

a) Just enter a title, seftitle and text into the form. [easiest]

b) Include a php-file instead of entering text into the textarea e.g.:

```
[include]./schemes/qimanfaya/intro.php[/include]
```

In the latter case you have to give the full path to the php-file (from your webserver root). In your php-file you have access to all the globals from qmd-CMS and to the database – thus giving you full control over almost anything. [This is easy and simple]

c) Write your own article-object and enter the name of the php-file containing the article-object into the field labelled „Object Loader for Article“. In this case you do NOT have to enter the path. The object-loader is expected to be in the selected scheme-folder. [This is probably complicated]

6.3.2 Article properties

[Menulink]: Here you can *hook* your article to a specific menu. If the menu is selected your article will be displayed (amongst others if more than one article is hooked to the menu-item in question).

From Version 0.8.6 you can hook one article to several menus. There are two buttons additionally available: „Add new“ (Menu to article) and „Remove Menu“. **It is strongly dicouraged to use this feature.**

Explanation: The actual idea was that a specific (self-written) article object may be reused and instantiated from different menu-entries. Well, the article-object can access the URL and decide what to display - so there's nothing wrong with this. But when it comes to SEO the articles will result in exact the same SEO-content in the header of the HTML-page. So please consider this when using this feature.

[Category:] You don't have to select a category, but it will give your users more abilities to navigate through your site and find the content they are looking for.

[Scheme/Layout:] A Scheme has to be selected otherwise the default scheme is being used for the article. Note: if any article is displayed, always the scheme for the article is used and not the one that is the general scheme you've set up using administration/settings.

[Default article for scheme:] If this checkbox is checked this article is used as the default article for the scheme selected in [scheme/layout]. This article is displayed when either

+ an empty category is selected
+a menu-entry was selected that contains no articles (but can contain submenus)

Caution: While testing in an almost empty database this might be confusing cause this article is probably always loaded. If you have no article set up as default article for scheme you will get a message that there is no content yet available. This might help you getting started.

[DC-Type] Should always be text (please refer to dublincore.org if in doubt)

[Meta-Doc-Type] Should always be XHTML-strict (please refer to w3c.org if in doubt)

[Object Loader for article]: is only used when you have written your own article-object. In this case it should be just the name of the php-file with your article-object without any path (the path from the scheme selected above is used automagically).

[Extra-Value for Object]: if you have your own article object you can save here some additional info. This enables you to re-use your object and have some information from which article it was actually coming.

[show in subcategories]: Determines if the article is shown when in Category overview mode of the [category]

[show on home page]: If checked the article is shown on th entry-point of your site (www/mysite.com/)

[Publish now]: Determines whether the article is available for users/published.

[show as a page]: do NOT show VAC-CONTAINERS.

[show this page as fullscreen]: The article contains all and everything. Nothing else is displayed. You have the css- and JavaScript-files loaded that are in your selected scheme-folder.

6.3.3 Preview

This is just a rough preview. The special css-files are not active so this might differ from the actual result.

The preview is not available if you have your article included or if there is an object-loader defined.

6.3.4 object specific data

This is always empty for default- or included articles. If there is an object-loader defined AND you have written an editor for your article-object the editor is displayed here. This makes it easy to maintain completely different articles (probably on a different database) within the same administration tool for articles.

6.3.5 Customize

The next is for SEO:

[Keywords-Meta]: The keywords for this article, so Search-Engines are able to find the content. While typing the keywords you see the number of characters used.

[Description-Meta]: The intro to the text which is displayed in the results that Search-Engines collect. While typing the description you see the number of characters used.

[Description/Subject (META)]: A short descriptive Text about the content.

The next items are related to the *behaviour*:

[Display Title]: Whether or not the title will be displayed. this gives you the option to have an internal title within the CMS and another real title that is shown to the user (in your article) – or no title at all.

[show date to this article]: Whether or not the creation date be displayed.

[comments]: You have several options to handle commentability for each article:

- comments off
- comments frozen – old comments are shown – no new comments allowed
- commentable – commenting allowed for everyone with a valid e-Mail-adress
- show frozen comments only for users – only registered users can see comments. Further commenting is disabled.
- commentable for users only. Only users can post comments.

  NOTE: Enabling of comments is only applied if you have comments enabled in your system-settings

[Publish date]: shows the time on the server and for a new article it defaults to the current time.

If you chose a future posting just enter the time from which on the article should be displayed. In article overview for admins and mods the article is displayed anyway and the 'future posting' is highlighted. So you can easily identify these articles.

Note: there is no DRAFT-MODE. Use future posting instead.

6.3.6 Article-Languages

On the right hand side of the article-edit-box there are the available languages – those you have installed in your system. If you create a new article it is created in the system-default language. If you save it and re-edit the article you see here the links to '`create new for <lang>`' - where <lang> refers to all other languages installed and `@ANY` which refers to articles that don't need to be translated. The latter may be used if you only have an image in your article or other content that doesn't require translation.

If you click on '`create new for <lang>`' the article will be copied into another language. You have to do the translation for yourself. (-; The SEO-fields for Metadata are copied as well which makes translation easier. But don't forget to tranlate them as well!

If the article exists in another language the shortcut '`create new for <lang>`' is changed to '`edit in <lang>`'.

6.3.7 Making a copy of an existing article

At the bottom of the edit-article page you find a button that enables you to copy an article – just in case you need another version of the same article with all its parameters preserved. Don't confuse this with languages. The copy will have only the current language. If you want to create the article in another language click on the right hand top “create article in 'xx'” where xx stands for those languages where the article is not available yet.

6.3.8 Deleting articles

There are two options: Delete the article in the current language or delete all copies in all languages and the article itself in general.

6.4 Editing VACs

A VAC is a “Value-Added-Container”. A container that might add some value to your page.

VACs are defined in the index.php-file of the scheme-folder from which the main article is loaded.

In general a VAC is to be edited in more or less the same way, as described in Chapter 6.3Editing articles

Some features that do not apply to VACs are disabled and invisible and others are included such as the following checkboxes::

show always – this means, that the VAC-container is shown even if it is not related to the current category. if unchecked the VAC is only shown when it is in the same category as the main-article[2]

on left side - VAC-container is shown on the left side (if index.php includes it)

on right side - VAC-container is shown on the right side (if index.php includes it)

header VAC - VAC-container is shown on top (if index.php includes it)

bottom VAC- VAC-container is shown at the bottom (if index.php includes it)

[2]Try to check this if your VAC is not displayed

7 Customizing qmd-CMS

Before you customize the demoversion make a backup of the scheme: default.

1.) Your best bet is: just alter the css-file(s) overwriting the offending section instead of introducing new styles everywhere...

2.) Do whatever you like with the index.php

3.) You can write a handler for articles (a new article object). This one can load its own *.js and *.css-Files. But it is limited to display itself. Either in the center of your webpage with all the headers and ads at the bottom (as a default).

- or you assign a simple index php to an otherwise not used scheme where you only call the **`center`** function in index.php....

```
<?php

    center();

?>
```

In this case you would only see what your article object is displaying. It could make up its own headers and footers...

7.1 Debug

In order to alter something you might want to debug your code – if it doesn't behave as you expected.

7.1.1 Controling debug-output

In order to determine if you are in test mode there is a variable (actually a constant) that is located in **`qmd_const_settings.php: __y_local_test`**

This variable is set to 1 if the TLD of your website is set to **`.test`** – e.g. www.qmd.test

It is safe to do some debug output if **`__y_local_test==`** 1.

In order to refine control over debug-output there is another variable in the same file: **`__y_DEBUG`**.

`__y_DEBUG` is set to 0 (zero) by default in a live environment and to 3 in any test-environment. You can enter different values.

All I got to offer may be NOT state of the art when it comes to debugging websites – but at least it works for me. So if you need some debug output you might use one of these:

7.1.2 Debug functions

There are debug functions implemented in order to track down some misbehaviour:

- msgbox('Hello'); shows an alert from JavaScript – with some trace
- yecho($foo); shows an object $foo in detail (on the site you're testing)
- debuglog($msg); logs $msg to a file (this is better for live mode)[3]

Just insert (one of) them to get some output and see what's inside of some $variable.

For live environments none of these is applicable...

You might consider using this:

```
send_sys_message($Error_Info,$MoreInfo);
```

This will send a message with Subject $Error_Info and Body $ModeInfo to the admin including a complete call-stack and the URL in question that caused the error. Next time you log in you will find a message in your inbox.

If something goes utterly wrong you should use qmd_die($errorMessage).This also sends a system message to the admin.

7.1.3 test your functions

In order to check a newly introduced function (or object) you should check the results with some sensible input. In test – mode you have the function <http://www.myqmd.test/qtest>

So the suffix will call a (F) named qtest() which resides in ./qMDCMS/qmd_test.php

Make a call to the function you want to test and give it some sensible data. Check the results against what you've expected with the functions in chapter Debug functions above.

7.2 Initialization

The initialization process is described in qmd_notes.ods (in sheet: „structure“) which should accompany your version of qmd-CMS.

[3](F) debuglog() will only be logging data in test-mode when __y_DEBUG is greater than 1

7.3 User

The user is seeing the website that is delivered to him/her.

The URL contains (usually) as a last part the URL-friendly Title of the webpage.

The general layout of the website that the user sees is that of the

- index-page in the scheme-folder of the article
- the css files in this folder

The content to be displayed (Header, Menu-Bar, Value-added containers … Ads) is defined by what the index.php – file is doing.

The actual content the user has requested is the article and usually you'll see the article displayed in the center.

7.4 Any Language

Sometimes your article may be available in any language. Instead of changing the language and inserting an article per copy and paste you may just have one version of the article in order to keep it easily maintained.

Well, use:

```
UPDATE articles_ml set lang = '@ANY' where id=#ID;
```

Just replace the '#ID' with the id of the article in question. So you will get rid of the notion that the article is not available in your language.

This may be useful for VACs that just have an image with a link and don't need to be displayed in several languages.

8 Programmers-Section

If you are about to use the code, please read the License carefully.

Additionally I ask everybody to adhere to some standards, at least if you are going to merge your extensions/alterations to the main branch of this software:

8.1 Reserved names

8.1.1 logoff

I prefer the term “logoff” instead of logout.

Sorry, but it is a reminiscence regarding the **C**ontrol**P**rogram [`cp`] at `profs` which was the IBM-**Pr**offessional-**Of**fice-**S**ystem in the eighties of the last century.

I kinda contributed to that one too.

I hope fellow hackers using this code will respect it and keep it alive.

9 Appendix