Department of Technology, Mathematics and Computer Science

DEGREE PROJECT
2004:DS21

Creating a modular Content Management System

Jukka-Pekka Luukkonen

HÖGSKOLAN
TROLLHÄTTAN • UDDEVALLA

DEGREE PROJECT

Creating a modular Content Management System

Jukka-Pekka Luukkonen

Summary

This report is a ten credit degree project documenting the creation of a lightweight, modular Content Management System (CMS) from planning to implementation and testing. The report is aimed at programmers and people who are interested in web standards and what a CMS is.

A CMS can be used by an organization to efficiently manage and distribute information.

The goal of the project was to create an Application Programming Interface (API) and software classes for use in the CMS. Cascading Style Sheets (CSS) are used in the design of the layout and in the HTML output to separate content from presentation and making the HTML more compact.

The programming language used is PHP together with the free web server Apache.

Publisher:	University of Trollhättan/Uddevalla, Department of Technology, Mathematics and Computer Science, Box 957, S-461 29 Trollhättan, SWEDEN Phone: + 46 520 47 50 00 Fax: + 46 520 47 50 99 Web: www.htu.se		
Examiner:	Stanislav Belenki		
Advisor:	Dag Kihlman		
Subject:	Computer Science	**Language**:	English
Level:	Advanced	**Credits**:	10 Swedish, 15 ECTS credits
Number:	2004:DS21	**Date**:	August 28, 2004
Keywords	CMS, XHTML, CSS, PHP, MySQL		

DEGREE PROJECT

Preface

I would like to thank Stanislav Belenki of the University of Trollhättan / Uddevalla for his assistance and Péter Tihanyi for help with PHP programming problems.

Contents

Appendices

List of symbols

API Application Programming Interface – A set of commonly used functions provided by a system for developing applications.

CMS Content Management System – A system for collecting, organizing and distributing information.

CPU Central Processing Unit – The part of a computer or other hardware device that interprets and executes software instructions from memory.

CSS Cascading Style Sheets – A method to style a document written in HTML.

HTML Hypertext Markup Language – A markup language for creating web pages.

MD5 Message-Digest algorithm 5 – A hashing algorithm used to calculate checksums from data.

PHP PHP: Hypertext Preprocessor – A programming language for creating web applications.

XHTML Extensible Hypertext Markup Language – An evolution of HTML by mixing HTML with XML to create a modern markup language for creating web pages.

1 Introduction

A definition of a content management system is according to the Wikipedia website [1]: "A content management system (CMS) is a system used to organize and facilitate collaborative content creation."

A CMS is software that is used for storing, presenting and distributing an organization's information. Often an organization may have a static website to distribute information. That means that each page of the site is created manually by the staff using for example an HTML editor such as Macromedia Dreamweaver. This method works well for a while, but eventually the size of the site will grow and information will be hard to track and maintain. In such cases a CMS can help the organization to collect the information in a uniform and managed way for easier distribution and maintenance. The benefits of using a CMS are for example greater consistency in the information (content is formatted using the same syntax), faster publishing of information (as soon as a user submits content it can be made visible online), decentralized authoring (a user may submit new information from anywhere via a client application or a web browser).

The information contained in a CMS may be used only internally in the organization or both internally and externally like for example in the case of a newspaper that uses the CMS for editing their articles and also making them available on a website for readers. The CMS can be an application to be run natively on an operating system or web based, where the interaction between it and the user occurs via a standard web browser.

An example of an organization that might use a CMS is a newspaper where the authors submit articles, pictures and other material for publishing. The articles are written in a special syntax using a mark up language, perhaps similar to Hypertext Markup Language (HTML) to style the text (setting bold, italic or underline style on text for example). The advantage of this method is that all the content is written using the same mark up language so that the CMS or another application can be responsible for how the content is interpreted and presented. Besides being used for publishing of plain articles the CMS can have modules for added functionality such as a discussion forum where readers may discuss or comment on each article.

There are many free web based CMS's (such as Php-Nuke [2]) available for download on the Internet but one of the differentiating goals of the one in this degree project was to make it adhere to the web authoring standards XHTML and CSS (Cascading Style Sheets) and to try making it lightweight for lesser capability hardware devices.

2 Purpose

The purpose of this degree project is to create an Application Programming Interface (API) for use in a CMS or other web applications that need access to databases. The output will conform to modern web standards such as XHTML and CSS to ensure the CMS' output is compact and that it works with as many hardware devices as possible.

3 Limitations

To limit the area of this degree project I focused on creating the API, a select few modules for testing the API in a CMS environment and using relevant technologies to make its output HTML modern web standards compliant. The API can then be used to extend the CMS' with additional modules beyond my initial work.

4 Method

This project was mainly a practical one. Most of the needed facts can be found on the Internet so that was the main source of information. The project was done using the waterfall approach in these steps:

- Found out system requirements
- Decided on software to use
- Planned the API and needed components
- Programmed the API and components using a text editor
- Created a test module for the CMS
- Tested the CMS in different browsers to ensure that it is usable

An important part of this project was also to study CSS and create test mock-ups to find out how proper webpage layout by only using CSS is done.

5 Background

5.1 Web standards

The World Wide Web Consortium (W3C) was created in 1994 to develop standards for protocols on the World Wide Web (WWW). The consortium consists of around 350 member organizations that work together creating the standards. Two important standards which are continuously being worked on are HTML and CSS (Cascading Style Sheets).

5.2 HTML

HTML is a markup language used to for displaying information online on the World Wide Web. HTML was created as an application of SGML (Standard Generalized Markup Language), a universal language created for marking up documents using so called tags. A tag is a string of text enclosed between < and > characters, for example <tag>. For each tag an end tag is required in the format </tag>, this tells the parser of the document that the block of information for that tag has ended.

The HTML is written to a text file where the information the user wants to display is contained within tags that describe for the client program (typically a web browser) how the information in the HTML file should be presented, like for example using bold, italic or underlined text, where paragraphs begin and end and so on. Because the HTML specification was released in several versions as the standard evolved, the enforcing of how writing HTML documents should be done became difficult. HTML also has limitations such as that it is not possible to write all sorts of documents with it easily, for example mathematical and musical notation is difficult to write in HTML. This lead the W3C to find a solution to the problem of making a markup language that suits every need.

5.3 XML

XML (eXtensible Markup Language) is a meta language created by the W3C to define rules for new mark up languages. Using XML a new mark up language can be created for any application needed. For example when designing a mark up language for storing information about a DVD movie collection you might want tags called <disctitle> and <year>. A Document Type Definition (DTD) is a text file containing rules that say which tags are allowed in the XML document. By referencing a DTD from the XML document it is then possible to verify that the XML document is valid and well formed by using validating software. The W3C used XML as a basis for creating the successor of HTML called XHTML or Extensible Hypertext Markup Language.

5.4 XHTML

XHTML is the W3C's answer to creating a new markup language for web content without breaking compatibility with the old ways of creating web pages. XHTML comes with three different DTDs called strict, transitional and frames. Transitional is the most used currently as it includes many of the old HTML tags that many people still want to use. The frames DTD is similar to the transitional DTD but caters for the document body containing elements for the different frames the web page should contain. Finally, the strict DTD does not allow any styling tags in the HTML itself, all styling must be placed in a CSS file. The XHTML standard clearly defines how tags should be written so interpreting the files becomes easier for software. For example in XHTML all tags and their attributes must be written in lower case characters and even single tags like
 must be ended using a slash like so
 unlike in HTML.

The terms HTML and XHTML will be used interchangeably in this degree document.

5.5 CSS

A simple explanation of what Cascading Style Sheets does is offered on the W3C website:

"Style sheets describe how documents are presented on screens, in print, or perhaps how they are pronounced. By attaching style sheets to structured documents on the Web (e.g.

HTML), authors and readers can influence the presentation of documents without sacrificing device-independence or adding new HTML tags." [3]

By putting the layout in separate CSS files that are included into HTML files it is now possible to separate the content that the author wants to present from its presentation. There are several advantages of this system compared to the old method of using tables. Using tables for layout works, but it was not meant to be used for that purpose. It causes a heavier download for the client machine because complicated layouts use lots of <table>, <tr>, <th> and <td> tags. In addition, table heavy web pages require more processing power to render on screen.

When using layout by CSS the layout commands may be placed in one separate CSS file that is included into several different HTML pages. This makes it possible for the client machine to cache the CSS file and thereby reduce download times. HTML in combination with CSS makes the mark up a lot cleaner because the HTML doesn't contain any redundant styling tags. Reducing the download size of a page is very important because many smaller devices such as mobile phones and PDA:s have less memory, less processing power and in the case of mobile phones slower Internet connections than desktop computers do.

Here is a picture that demonstrates the usage of style sheets together with an HTML document:

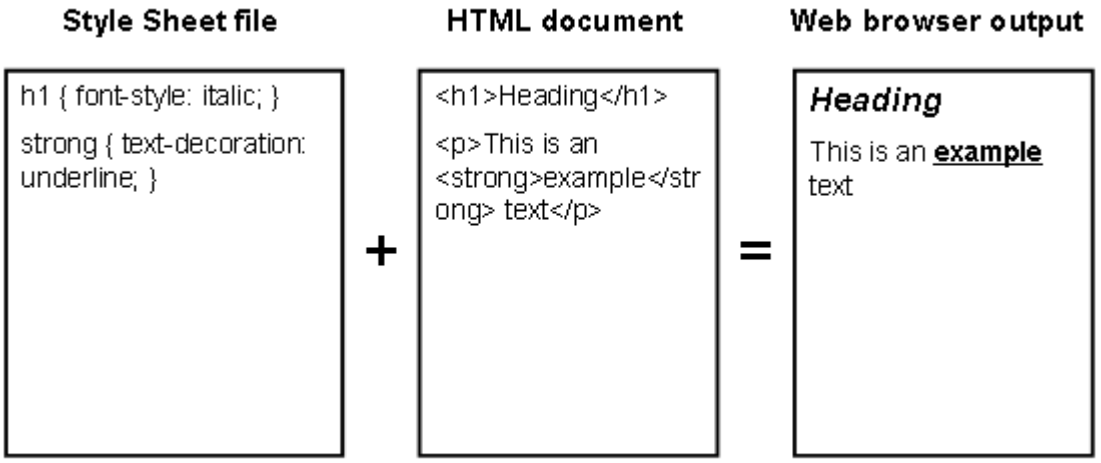

Figure 5-1 Including a style sheet into a HTML document

This picture makes the point of CSS removing redundant style tags in HTML clear. To change the style of several HTML documents the changes now only need to be done once; in the CSS file.

5.6 PHP

PHP (PHP: Hypertext Preprocessor) is a programming language used for developing dynamic web applications. In syntax it reminds of the C programming language and it is often used as a structured programming language although it has basic object oriented functionality. PHP is open source and thereby the source code for it is free for anyone to download, modify and use. It has support for many different web servers but the Apache web server is the one it is most often used with.

6 Development of the system

6.1 Initial planning

Before starting development of a larger system some planning is needed. In this case it required coming up with some generic features a potential user or system customer might want because there was no specific user who wanted a system.

Important requirements that were considered were:

- Multi language support
- System portability
- Extensible for additional modules
- Easy to modify the look and feel of the system

Multi language support might for example be needed by some company that wants to have a site made available for their customers in several languages.

The programming language selected for implementing the system was PHP [4] which is a popular language for creating web applications. One of PHP's main benefits is cross platform availability which makes it possible to move the system from one operating system to another easily. PHP support can also easily be added as a component into the free web server Apache [5] which is used in this project. The current stable version of PHP (versions below 5.x) unfortunately does not include full object oriented capabilities

such as private variables and methods but in this project that did not cause any problems.

To make it easy for users to create their own modules an API with helpful functions is needed. Such functions include permission control, multi language functionality and so on.

Thanks to using CSS the CMS is just by that already quite easily customizable by users. By changing the settings in the CSS file the user may change colors and add decorative graphics to their site.

The development was done on a computer with Windows 2000, Apache 2.0.44, PHP 4.3.1 and MySQL 3.23.55.

6.2 Code standards

To make the source code of the CMS more readable and uniform some standards for programming were devised. The following is a list of standards used in the CMS project:

6.2.1 Variable naming

- Normal variables: $thisIsAVariable
- Private class variables: $_thisIsAPrivateVariable (even though PHP versions below 5.x do not support true private variables this naming principle was chosen for the sake of clarity.)

6.2.2 File naming

- Normal code files: file.php
- API class files: class.classname.php

6.2.3 Database naming

- Table naming: Users, GroupMembers
- Column naming: mycolumn

6.2.4 Source code standards

- All code, names and comments should be written in English.
- Function and variable names will be named as mySmallFunction() and $numRows unless it is a short lived counter variable or similar in which case names such as $x are acceptable.
- Constants and variables not meant to be changed will be named in uppercase as MYCONSTANT or $MYCONSTANT.
- Tabs will be used to indent the code, not spaces unless for some special purpose such as making a simple ASCII illustration.

6.3 Directory structure design

To keep the file hierarchy of the CMS source code clear and easy to maintain, files are grouped in their own subdirectories.

```
/
 index.php
   └modules/          Subdirectory for modules
      └admin/            Admin module subdirectory
         └tpl/              Templates for admin module
            └se/                Templates for Swedish language
       article/          Article module subdirectory
         └tpl/              Templates for article module
            └fi/                Templates for Finnish language
```

Figure 6-1 Directory structure example

In the directory modules/ there is a subdirectory for each module. In each module's subdirectory there is a directory called tpl where template files for the module's default language should be placed. For additional languages the templates can be placed under a subdirectory named after a country's two letter acronym.

6.4 API and component design

Once the basic requirements were established it was time to start the planning and designing of the API with which modules for the CMS are constructed. In traditional PHP programming the scripts themselves are mixed with HTML tags. For a programmer it is not difficult to see what is what but because one of the requirements was for the system to be easily modifiable a different method was needed. To solve this problem a template class was constructed. By using that method someone who only knows HTML can easily change the look and feel of the CMS without having to worry about ruining some code section by mistake. For experienced programmers the possibility of working with shorter (not cluttered with HTML code) pieces of source code should also be beneficial.

By using PHP as programming language the system is already portable to other platforms to a large degree. The CMS however stores most of the data in a database which may not be available on the operating system the user wants to migrate to. Even though PHP has built in support for many different database vendors a method for abstracting the database access was needed to make the system truly portable. For this purpose a class for abstracting database access was designed, essentially it wraps the calls to the different database functions into common function names.

The final component of the system is an include file with commonly used functions for things like checking access permissions, login status and such.

Here is a basic illustration of the whole system:

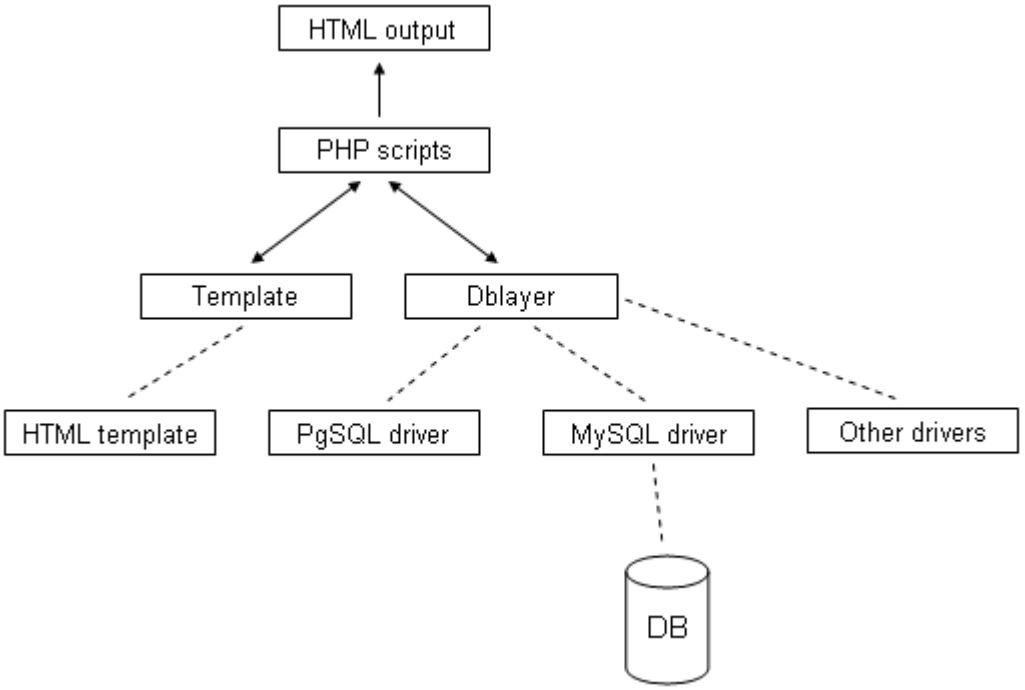

Figure 6-2 System components

6.5 Component interaction

Here is a detailed picture that illustrates how the CMS uses the components to produce output for a web browser:

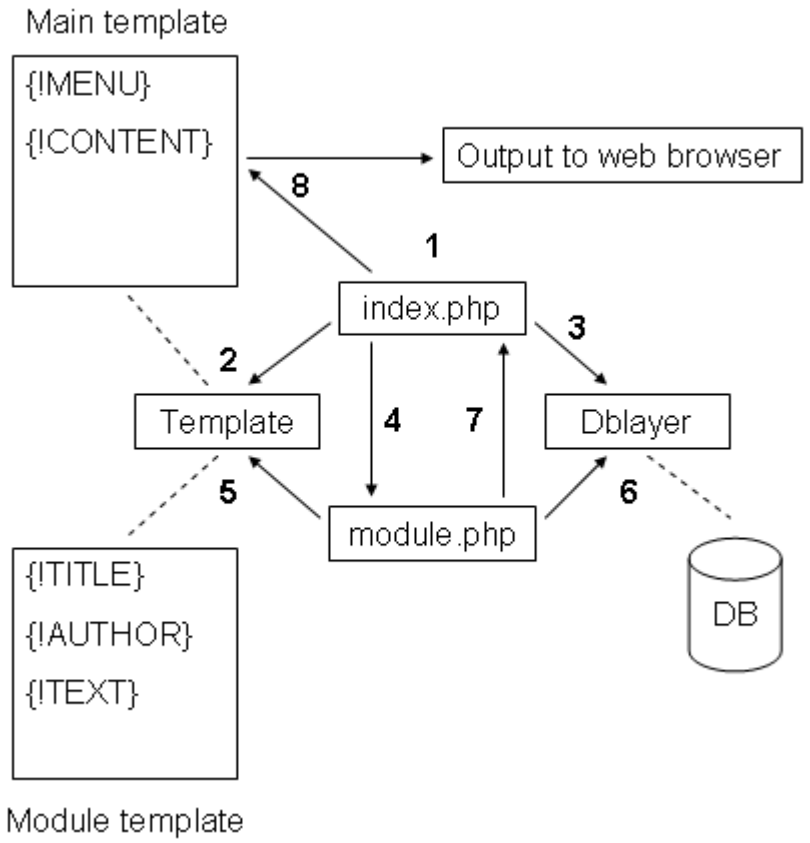

Figure 6-3 Component interaction to produce output for a web browser

What happens during the scripts' execution is the following:

1. index.php loads the classes for the classes Template and Dblayer along with the header API file.
2. index.php instantiates an object of Template and loads the Main template into it.
3. index.php instantiates an object of Dblayer and opens a connection to the database.
4. If the user has selected a function, a module is loaded and its object is instantiated.
5. The module loads its own template.

6. The module gets the data it needs from the Dblayer object that index.php created.
7. The module fills its template with the data from the database and returns the resulting output to index.php
8. index.php places the content it received into the main template, parses it and sends the output for the client web browser.

6.5.1 The Template class

The template class is used for loading, manipulating and displaying templates for web pages. The benefit of this is that the code and the layout of the pages are separated. The basic function of the template class is that when its constructor is called (when a new instance of the class is created) it loads the template the user wants. When the user calls it like:

```
$t=new Template("./modules/article/tpl/main.tpl");
```

the script first checks if the user has a cookie with a preferred language set. If there is, it tries to load the template from:

```
./modules/article/tpl/PREFERRED_LANGUAGE/main.tpl
```

If that fails it tries to load a template using the site wide constant DEFAULT_LANGUAGE like so:

```
./modules/article/tpl/DEFAULT_LANGUAGE/main.tpl
```

If that also fails the last option (which should never fail) is to load the file for the module's own default language from:

```
./modules/article/tpl/main.tpl
```

Usage instructions for the Template class can be found in appendix B.

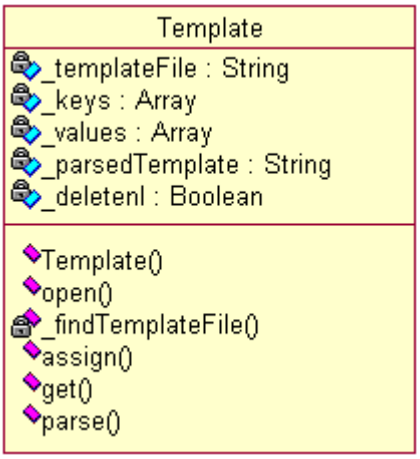

Figure 6-4 Template class diagram

6.5.2 The Dblayer class

The Dblayer class is used to abstract the database accesses from the CMS scripts. The object is instantiated as follows:

```
$db=Dblayer("mysql");
```

where the parameter is the name of the database driver. In this CMS only the MySQL driver is included and used, but writing additional ones shouldn't be difficult. Besides the usual methods for open, closing and querying a database the class also includes some useful methods for web development purposes. Included in the driver is the result set class Rs of which an instance object is returned for each executed query. The CMS uses a global variable called $db which is instantiated at the beginning of the index.php file, so all modules must access the database using that variable.

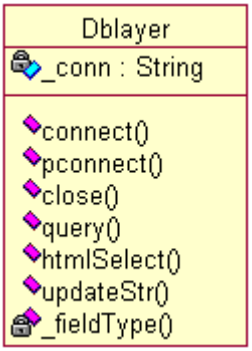

Figure 6-5 Dblayer class diagram

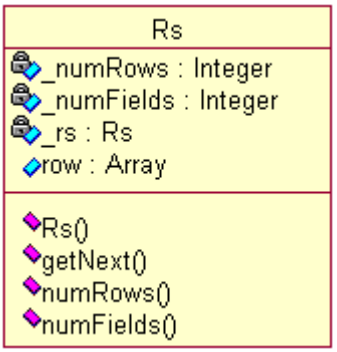

Figure 6-6 ResultSet class diagram, used by Dblayer class

6.6 Additional system design

After having the components to start creating the system itself it was necessary to design among others the permissions system for how user groups could be limited access to modules.

6.6.1 Permissions

After some consideration the idea being used is to have a system where a table in the database contains a row for each group that is given permissions for a particular module. An example row in the table looks like this:

ModuleID	GroupID	Permission
1	10	3

Figure 6-7 Example row for permissions table

The ModuleID refers to another table, for example 1 refers to the module article. GroupID refers to a group name, in this case Admin. The Permission column is a value with which a bit mask of permissions can be checked to see if this user's group has access for that operation on that module. The permission bits available are as follows:

Permission	Value
READ	1
WRITE	2

DELETE	4
CREATE	8
MODIFY	16

Figure 6-8 Permission bits

When a script needs to see if a user has access for a particular function on a module it can call the global function hasAccess("ModuleName", BITMASK). BITMASK is a combination of OR:ed module permissions (READ,WRITE etc.). The function will scan all the user's groups for those permissions and return true if the user has permission or false if not. Members of the Admin group always by design have access to everything. This CMS doesn't support user level permissions.

To set permissions for a module the module has to be activated from the admin panel.

6.6.2 Passwords using MD5 (Message-Digest algorithm 5)

MD5 was developed by Professor Ronald L. Rivest of MIT. An explanation of what MD5 does is given on the unofficial MD5 homepage:

"[The MD5 algorithm] takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA." [6]

The hashes are represented by a string of 32 hexadecimal digits, for example like: d41d8cd98f00b204e9800998ecf8427e which represents the input value of an empty string. MD5 hashes are commonly used to make sure that files that are distributed via the Internet have not been modified for some reason (data transfer errors, hackers trying to spread viruses etc). After downloading a file the user can verify if the received file's MD5 hash matches the one on the server end to ensure the file has been properly transmitted. MD5 can also be used to obfuscate a password so that it is not stored in its clear text form.

The login passwords in this CMS are stored as MD5 hashes which should be adequate in smaller non safety critical sites, for example most web bulletin boards use that system.

The side effect of using MD5 hashing is that there is no way of getting the password back if a user should forget the password. If that happens an administrator needs to set a new password for the user.

6.7 Database design

The system has five required tables; Users, Groups, GroupMembers, Modules and ModPerms. The following illustration shows the most important key relations between the tables (all column names are not shown, see Appendix C for full table definitions):

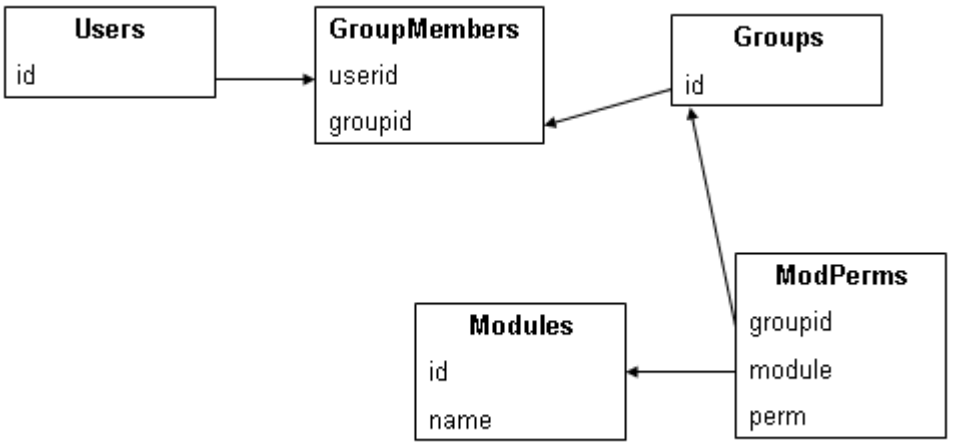

Figure 6-9 Database table relations

GroupMembers is a table added so that a user can be a member of several groups. The CMS uses a variable called $DBP which is added in front of every table name in queries, by doing this it is possible to have several CMS systems in one database. The difference between the systems would then just be the prefix before the table names.

6.8 Graphical User Interface design

The graphical design is an important aspect for the usability of a website. For this CMS a very common and proven standard design was used:

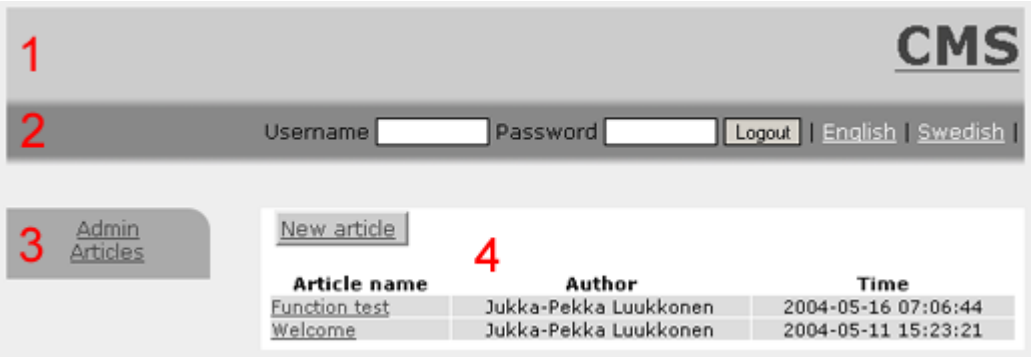

Figure 6-10 GUI design

1. Top logo area. Here a logo for the website / company / community can be placed.
2. Top menu. Used in this case for login / logout and language selection.
3. Left menu. The available modules are listed here.
4. Content area. The results from the select operation from a module are displayed in this area.

The gradients and other decorations are achieved by only using CSS. This is beneficial because the CSS file contains a section using the CSS @media handheld directive without those decorations for less capable clients such as mobile phones. This method reduces the download size and CPU usage for such clients. All the templates used were written in XHTML and styling was kept in the separate CSS file.

6.9 Module structure

To try the API a module for editing, posting and displaying text articles was made. For each article there are three different statuses possible; NORMAL, LOCKED and INVISIBLE. For everyone but members of the Admin group and logged in users belonging to groups with proper rights articles marked INVISIBLE are not available for

reading. The status LOCKED is used for locking a document so that two users may not try to edit it simultaneously.

Another module named admin was also created. This is used for the management of users, groups, permissions and modules for the site.

6.9.1 A module's general code structure

```
<?php
// Include the correct lang.php for this module
include dirname(__FILE__) . "/../../inc/loadlangfile.php";

class Module
{
 // main() is a required entry point function for all modules
 function main()
 {
       global $MODLANG,$DBP,$db;     // Commonly used globals
       $modResult="";    // Resulting output for this module
       switch(@$_GET["modop"])
       {
             case "operation1":
                   // do something
                   $modResult=$myOutput;
                   break;
             case "operation2":
                   // do something
                   $modResult=$myOutput;
                   break;
             default:
                   // Display default page
                   $modResult=$myOutput;
                   break;
       }
 return $modResult;      // Return resulting output to index.php
 }
}
```

This is the structure used in both the admin and article modules and while not being a strict requirement (except the need for a main() method) this structure has proven to be a working one.

6.10 Browser compatibility test

It is not enough to try a website for compatibility with Microsoft Internet Explorer even though it is the market leader. Browsers which are released in new versions more frequently such as Mozilla Firefox and Opera are gaining popularity and feature faster and more accurate rendering of web pages than Internet Explorer does.

The first test browser was Microsoft Internet Explorer 6.0 on which everything worked fine. After that it was tested with Opera 7.23 and Mozilla Firefox 0.8 and was found to be working properly in both browsers with minor differences in look. A small problem with combining the CSS background directive was discovered and corrected quickly. Viewing and using the system with the old Netscape Navigator 4.8 worked, even though the lack of proper CSS support in it made the page look quite bad. The system was also test viewed on a Nokia 3200 mobile phone, thanks to the use of CSS adapted for handhelds it is readable even on its 128x128 pixel display.

7 Conclusions

The goal was to create a CMS using a custom made lightweight API, following web standards such as XHTML and CSS and catering for hardware client devices of varying capabilities. The goals were met; the result is working as intended and was verified to work with different browsers.

The CMS can be quite easily extended with additional modules and its look can be changed by modifying the CSS file or the template files. As a plus some of the components of the API are generic enough that they may also be used in other web application projects.

During the planning and development the biggest hurdles were how to create the permissions system and the loading of templates depending on the preferred language of the user.

Further additions to the system might be for example to add user level permissions management to get finer control of permissions for modules. Another addition might be to perform the MD5 hashing by a JavaScript at the client side, although this means that a client that doesn’t support JavaScript can not login anymore.

References

1 Wikipedia, the free encyclopedia

Available: http://en.wikipedia.org/wiki/Content_management_system [2004-07-22]

2 PHP-Nuke website [Electronic]

Available: http://www.phpnuke.org/ [2004-07-22]

3 W3C Web Style Sheets website [Electronic]

Available: http://www.w3.org/Style/#what [2004-07-22]

4 PHP Hypertext Preprocessor website [Electronic]

Available: http://www.php.net/ [2004-07-17]

5 The Apache Software Foundation

Available: http://www.apache.org/ [2004-08-14]

6 MD5 Homepage (unofficial)

Available: http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html [2004-07-24]

A Reserved keywords

$db	$DEFAULT_LANGUAGE	$DBP
$DBUSER	$DBPASS	$DBNAME
$DBHOST	ADMINGROUP	READ
WRITE	DELETE	CREATE
MODIFY	$_COOKIE[”userlang”]	$_SESSION[”userid”]
$_SESSION[”usergroup”]	$main	$result
$status	$statusMsg	$result
$_GET[”op”]	$_GET[”modop”]	

B API function reference

General functions

```
function isLoggedIn()
 Check if user is logged in
 Returns:
       true if user is logged in, else false


function isAdmin()
 Check if user is an admin
 Returns:
       true if user is admin, else false


function hasAccess( $mod, $access )
 Check permissions
 Parameters:
       $mod: module to check for
       $access: bitmask of access permissions
 Returns:
       true if user has access, else false
```

Class: Template

Usage instructions:

Keys and values

The template that is loaded can contain key markers, these are in the format {!KEY} or {#!KEY}. When a user wants to replace these markers with some data he can call the

assign(“!KEY”, “My value”) method to assign a value for the specified key. There are two types of keys, the regular one which looks like {!KEY} and one like {#!KEY}. The latter one is replaced by the its value at all occurrences in the template at parsing, this functionality was included so that the user doesn’t have to assign the same value several times if he wants to copy a value several times in the template.

The TPLREP directive

If the user wants to repeat some piece of information, such as listing username and password pairs on a webpage the TPLREP directive is handy. By enclosing the text section to be repeated in <TPLREP !USER> {!USER} {!PASSWD} </TPLREP !USER> tags the user’s script may assign as many values to !USER as are needed. A call to parse() will assign them in order and repeat that section for as many times as there are values for !USER.

The TPLSECTION directive

If the user wishes to divide a template into sections, that is also possible by enclosing each section with the tags <TPLSECTION sectionname> template content </TPLSECTION sectionname>. When using the parse() and get() function the user must then pass the section name as a parameter to get the correct one parsed or returned.

Note that even though it is possible to have several sections in one template object the key and value pairs are stored in the same object so it is not possible to have two keys with the same name in different sections.

Example template

example.tpl

```
{!TITLE}
<TPLREP !ID>
Number: {!ID} Name: {!NAME}
</TPLREP !ID>
{#!HELLO} {#!HELLO}
```

Example source code

example.php

```
1       include(“./inc/class.template.php”);
2       $t=new Template(“./example.tpl”);
3       $t->assign(“!TITLE”, “Example page”);
4       for($i=0; $i<5; $i++)
5       {
```

```
6             $t->assign(“!ID”, $i);
7             $t->assign(“!NAME”, “User”.$i);
8       }
9       $t->assign(“#!HELLO”, “Repeated hello”);
10      $t->parse();
11      echo $t->get();
```

Line by line description:

1 Include the Template class file.

2 Create new object, load template example.tpl into it.

3 Assign string value “Example page” to key !TITLE.

4 Loop 5 times.

6 Assign number to key !ID.

7 Assign string User + number to key !NAME.

9 Assign repeated string value “Repeated hello” to key #!HELLO.

10 Parse template, replacing keys with their respective assigned values.

11 Display result.

Functions:

```
function Template($tplFile, $delnl = false)
 Loads template file.
 Parameters:
       $tplFile: path to template file
       $delnl: set true to delete newlines from parsed result


function open($tplFile, $delnl = false)
 Manually loads template file.
 Parameters:
       $tplFile: path to template file
       $delnl: set true to delete newlines from parsed result


function _findTemplateFile($path)
 Finds correct template depending on language.
 Parameters:
       $path: path to template
 Returns:
       correct path to template


function assign($key, $value)
 Assign keys and values for later parsing
 Parameters:
       $key: key to assign
       $value: key's value
```

```
function get($subTemplate = “”)
 Return contents of specified parsed template
 Parameters:
       $subTemplate: wanted section
 Returns:
       parsed template


function parse($subTemplate = “”)
 Replaces assigned keys with their values in the templates in
 specified subtemplate
 Parameters:
       $subTemplate: section for parsing
```

Class: SQL

Functions:

```
function connect($db, $user, $pass, $host=”localhost”, $port=3306)
 Open database connection
 Parameters:
       $db: database name
       $user: database user
       $pass: database password
       $host: database host address
       $port: database port
 Returns:
       true on success


function pconnect($db, $user, $pass, $host=”localhost”,
$port=3306)
 Open persistent database connection
 Parameters:
       $db: database name
       $user: database user
       $pass: database password
       $host: database host address
       $port: database port
 Returns:
       true on success


function close()
 Close database connection


function query($qry)
 Execute query
 Parameters:
       $qry: SQL query
 Returns:
       new result set object


function htmlSelect($qry, $name, $allowNull=false,
$selected=false)
 Returns HTML <select> tags
```

```
 Parameters:
       $qry: SQL query for finding values to use in <option> tags
       $name: name of the <select> tag
       $allowNull: is a null answer allowed?
       $selected: which id is selected, if any


function updateStr($table, $arr, $where, $exclude="")
 Create SQL UPDATE string
 Parameters:
       $table: table to UPDATE
       $arr: array of values to SET
       $where: WHERE clause
       $exclude: values in $arr to exclude from SET
Returns:
       SQL UPDATE string built from parameters


function _fieldType($table, $columns)
 Find type of column
 Parameters:
       $table: table in database
       $columns: comma separated list of column names
 Returns:
       array with colname/coltype pairs
```

Class: Rs

Functions:

```
function getNext($mode=DBL_BOTH)
 Advance result set pointer to next row
 Parameters:
       $mode: mode for returning data (DBL_BOTH, DBL_ASSOC or
       DBL_NUM)

function numRows()
 Get number of rows in result set

function numFields()
 Get number of columns in result set
```

C Database tables

```
CREATE TABLE Articles(
  id int not null auto_increment,
  lang char(2) not null,
  author int not null,
  articlestatus int not null,
  createtime datetime not null,
  lastedit datetime,
  articlename varchar(255) not null,
  content text not null,
  primary key(id)
);

CREATE TABLE Users(
  id int not null auto_increment,
```

```
  username varchar(50) not null,
  passwd varchar(50) not null,
  fullname varchar(100) not null,
  email varchar(100),
  address varchar(100),
  zip varchar(10),
  city varchar(50),
  country varchar(50),
  phone varchar(50),
  mobilephone varchar(50),
  usergroup int not null,
  userstatus int not null,
  created datetime not null,
  lastlogin datetime,
  primary key(id)
);

CREATE TABLE ModPerms(
  module int not null,
  groupid int not null,
  perm int not null,
  primary key(module,groupid)
);

CREATE TABLE Modules(
  id int not null auto_increment,
  modname varchar(50) not null,
  unique(modname),
  primary key(id)
);

CREATE TABLE Groups(
  id int not null,
  groupname varchar(50)
);

CREATE TABLE GroupMembers(
  id int not null auto_increment,
  userid int not null,
  groupid int not null,
  primary key(userid,groupid)
);
```

D Source code

/index.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

  session_start();

  // Load language file
  if( @!include(
"./tpl/{$_COOKIE["userlang"]}/lang.{$_COOKIE["userlang"]}.php" ) )
    if ( @!include(
"./tpl/{$DEFAULT_LANGUAGE}/lang.{$DEFAULT_LANGUAGE}.php" ) )
```

```
      if ( @!include( "./tpl/lang.php" ) )
        exit("Error. No language file was found.");

  // Include important files
  include("./inc/class.template.php");
  include("./inc/class.dblayer.php");
  include("./inc/globals.php");
  include("./inc/functions.php");

  $db=Dblayer("mysql");

  // Connect to database
  if( !$db->pconnect($DBNAME, $DBUSER, $DBPASS, $DBHOST) )
    exit($LANG["STATUS_ERR_DB_CONNECT_ERROR"]);

  $main=new Template("./tpl/main.tpl");

  $result="";  // Output result from modules

  $status="";  // Status CSS
  $statusMsg="";  // Status message

  switch( @$_GET["op"] )
  {
    // Load mod and create object instance
    case "loadmod":
      $inc="./modules/{$_GET["mod"]}/{$_GET["mod"]}.php";
      include("./modules/{$_GET["mod"]}/{$_GET["mod"]}.php");
      $modObject=new $_GET["mod"];
      $result=$modObject->main();

      break;

    // Log in user and set userid in session
    case "login":
      if( isset($_POST["f_login"]) )
      {
        $sql ="SELECT id,usergroup FROM {$DBP}Users WHERE
username='";
        $sql.=$_POST["username"] ."' AND
passwd='".md5($_POST["password"])."'";

        $rs=$db->query( $sql );
        $rs->getNext();

        if( isset( $rs->row["id"] ) )
        {
          $status="status_ok";
          $statusMsg=$LANG["STATUS_LOGIN_SUCCEEDED"];
          $_SESSION["userid"] = $rs->row["id"];
          $_SESSION["usergroup"] = $rs->row["usergroup"];
        }
        else
        {
          $status="status_error";
          $statusMsg=$LANG["STATUS_LOGIN_FAILED"];
        }
      }
      else if( isset($_POST["f_logout"]) )
```

```
      {
        $status="status_ok";
        $statusMsg=$LANG["STATUS_LOGOUT_SUCCEEDED"];
        // Destroy session and unset cookie
        session_unset();
        session_destroy();
        setcookie("userlang", "", time()-3600);
      }
      break;

    // Set language cookie and redirect to mainpage to active new
setting
    case "setlang":
      setcookie("userlang", $_GET["language"]);
      header("Location: " . $_SERVER["PHP_SELF"]);
      break;

    default:
      $result=$LANG["WELCOME_MSG"];
      break;
  }

  // Display Admin in menu?
  if( hasAccess( "Admin", WRITE ) )
    $main->assign("!M_ADMIN", "<li><a
href=\"{#!PHP_SELF}?op=loadmod&amp;mod=Admin\">Admin</a></li>");
  else
    $main->assign("!M_ADMIN", "");

  // Display login or logout button?
  if( isLoggedIn() )
    $main->assign("!LOGINOUT", $LANG["BTN_LOGOUT"]);
  else
    $main->assign("!LOGINOUT", $LANG["BTN_LOGIN"]);

  $main->assign("!CONTENT", $result);
  $main->assign("!M_STATUS", $status);
  $main->assign("!M_MESSAGE", $statusMsg);
  $main->assign("#!PHP_SELF", $_SERVER["PHP_SELF"]);
  $main->parse();

  echo $main->get();

  $db->close();
?>
```

/inc/class.template.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

/*
* Template class
* Loads, manipulates and displays HTML template files
*/
class Template {

  var $_templateFile = array();    // Contains the sections of
templates
```

```
  var $_keys = array();        // Keys for variables to be
replaced
  var $_values = array();        // Values for variables to be
replaced
  var $_parsedTemplate = array();  // Parsed templates
  var $_deletenl = false;        // Delete newlines after parsing?

  /*
  *  Read .tpl file depending on language settings. Get the
section chunks
  *  out of it and assign them to $this-
>_templateFile($sectionName =>
  *  $sectionContents)
  *  Parameters:
  *  $tplFile: path to template file
  *  $delnl: set true to delete newlines from parsed result
  */
  function Template($tplFile=null, $delnl = false)
  {
    global $DEFAULT_LANGUAGE;

    // User wants to open a template manually
    if( $tplFile == null )
      return;

    // Find path to template file depending on language and
template
    // availability
    $path=$this->_findTemplateFile($tplFile);

    $fd=fopen($path, "rb");

    $templateTemp=fread($fd, filesize($path) );

    $templateTemp=ltrim($templateTemp);

    // Divide into separate sections
    $pos=strpos($templateTemp, "<TPLSECTION ");

    if($pos !== false)
    {
      if($pos > 0)
      {
        $default=substr($templatetemp, 0, $pos );
        $this->_templateFile[""] = $default;
      }

      $matches=preg_match_all("/<TPLSECTION
(.*)>([\S\s]*)<\/TPLSECTION \\1>/", $templatetemp, $tagparts);

      for($i=0; $i < $matches; $i++ )
      {
        $this->_templateFile[ $tagparts[1][$i] ] =
$tagparts[2][$i];

        $this->_parsedTemplate[ $tagparts[1][$i] ] =
$tagparts[2][$i];
      }
    }
```

```
    else
    {
      $this->_templateFile[""] = $templateTemp;

      $this->_parsedTemplate[""] = $templateTemp;
    }

    fclose($fd);

    $this->_deletenl = $delnl;
  }

  /*
  *  Read .tpl file, get the section chunks out of it and assign
them to
  *  $this->_templateFile($sectionName => $sectionContents)
  */
  function open($tplFile, $delnl = false)
  {
    $fd=fopen($tplFile, "rb");

    $templatetemp=fread($fd, filesize($tplFile) );

    $templatetemp=ltrim($templatetemp);

    $pos=strpos($templatetemp, "<TPLSECTION ");

    if($pos !== false)
    {
      if($pos > 0)
      {
        $default=substr($templatetemp, 0, $pos );
        $this->_templateFile[""] = $default;
      }

      $matches=preg_match_all("/<TPLSECTION
(.*)>([\S\s]*)<\/TPLSECTION \\1>/", $templatetemp, $tagparts);

      for($i=0; $i < $matches; $i++ )
      {
        $this->_templateFile[ $tagparts[1][$i] ] =
$tagparts[2][$i];

        $this->_parsedTemplate[ $tagparts[1][$i] ] =
$tagparts[2][$i];
      }
    }
    else
    {
      $this->_templateFile[""] = $templatetemp;

      $this->_parsedTemplate[""] = $templatetemp;
    }

    fclose($fd);

    $this->_deletenl = $delnl;
  }
```

```
  /*
  *  Finds correct template depending on language
  *  Parameters:
  *  $path: path to template
  *  Returns: correct path to template
  */
  function _findTemplateFile($path)
  {
    global $DEFAULT_LANGUAGE;

    // Try to find user's preferred language template?
    if( isset( $_COOKIE["userlang"] ) )
    {
      $pos=strrpos($path, "/");
      $filePath=substr($path, 0, $pos) . "/" .
$_COOKIE["userlang"] . substr($path, $pos);

      if( file_exists( $filePath ) )
        return $filePath;
      else
      {
        // It wasn't found so try to find the default language one
        $pos=strrpos($path, "/");
        $filePath=substr($path, 0, $pos) . "/" . $DEFAULT_LANGUAGE
. substr($path, $pos);

        if( file_exists( $filePath ) )
          return $filePath;
        else
          return $path;
      }
    }
    else
    {
      // Try to find default language template
      $pos=strrpos($path, "/");
      $filePath=substr($path, 0, $pos) . "/" . $DEFAULT_LANGUAGE .
substr($path, $pos);

      if( file_exists( $filePath ) )
        return $filePath;
      else  // Else return the module's default language one
        return $path;
    }
  }

  /*
  *  Assign keys and values for later parsing
  *  Parameters:
  *  $key: key to assign
  *  $value: key's value
  */
  function assign($key, $value)
  {
    $key="{".$key."}";
    array_push($this->_keys, $key);
    array_push($this->_values, $value);
  }
```

```
  /*
  *  Return contents of specified parsed template
  *  Parameters:
  *  $subTemplate: wanted section
  */
  function get($subTemplate = "")
  {
    return $this->_parsedTemplate[$subTemplate];
  }

  /*
  *  Replaces assigned keys with their values in the templates in
specified
  *  subtemplate
  *  Parameters:
  *  $subTemplate: section for parsing
  */
  function parse($subTemplate = "")
  {
    while ( preg_match("/<(TPLREP) (\S*)>/", $this-
>_templateFile[$subTemplate], $tagparts) )
    {
      // Make open- and closetags
      $opentag="<".$tagparts[1]." ".$tagparts[2].">";
      $closetag="</".$tagparts[1]." ".$tagparts[2].">";

      // Find out their positions in the template
      $tagopenstart=strpos($this->_templateFile[$subTemplate],
$opentag, 0);
      $tagopenend=$tagopenstart+strlen($opentag);
      $tagclosestart=strpos($this->_templateFile[$subTemplate],
$closetag, 0);
      $tagcloseend=$tagclosestart+strlen($closetag);

      // Make sure that the closetag exists
      if ($tagclosestart === false)
        exit(" Error in template, missing &lt;/TPLREP&gt; for
          variable ".$tagparts[2].".");

      // Get the part to be repeated
      $reppart=substr($this->_templateFile[$subTemplate],
$tagopenend, $tagclosestart-$tagopenend);

      // Find out how many times it is to be repeated
      $numreps=0;

      foreach($this->_keys as $k => $v)
      {
        if($v == "{".$tagparts[2]."}")
          $numreps++;
      }

      // Duplicate it N times
      $repptn=str_repeat($reppart, $numreps);

      // Get the complete string to overwrite
      $overwritestr=substr($this->_templateFile[$subTemplate],
$tagopenstart, $tagcloseend-$tagopenstart);
```

```
      // Overwrite it
      $this-
>_templateFile[$subTemplate]=str_replace($overwritestr, $repptn,
        $this->_templateFile[$subTemplate]);

    } // end while

    // Replace the strings
    $this->_parsedTemplate[$subTemplate] = $this-
>_templateFile[$subTemplate];

    for($i=0; $i < count($this->_keys); $i++)
    {
      if( $subTemplate != "")
        if( (strpos($this->_keys[$i], $subTemplate)) === false)
          continue;

      preg_match("/\{(.*)!(.*)\}/", $this->_keys[$i], $key);

      $str="";

      if( !strchr($this->_keys[$i], "#") )
      {
        $pos=strpos($this->_parsedTemplate[$subTemplate], $this-
>_keys[$i], 0);

        if ($pos !== false)    // And replace it with $v[1]
        {
          $str.=substr_replace($this-
>_parsedTemplate[$subTemplate], $this->_values[$i], $pos,
strlen($this->_keys[$i]) );
          $this->_parsedTemplate[$subTemplate] = $str;
        }
      }
      else
      {
        $str.=str_replace($this->_keys[$i], $this->_values[$i],
$this->_parsedTemplate[$subTemplate]);

        $this->_parsedTemplate[$subTemplate] = $str;
      }
    }

    // Delete newlines, tabs?
    if( $this->_deletenl )
      $this->_parsedTemplate=preg_replace("/[\r,\n,\t]/", "",
$this->_parsedTemplate);

  }
} // class Template

?>
```

/inc/class.dblayer.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

  /*
```

```
  * Dblayer class
  * Creates a database object for the specified database type and
returns it
  */

  // Dblayer constants
  define("DBL_BOTH", 1);
  define("DBL_ASSOC", 2);
  define("DBL_NUM", 3);

  function Dblayer($dbType)
  {
    include(dirname(__FILE__)."/dbdrivers/".$dbType.".inc.php");
    $o=new SQL();
    return $o;
  }
?>
```

/inc/dblayer/mysql.inc.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

  /*
  * Database class for querying, opening and closing connection
  */
  class SQL
  {
    var $_conn;

    function SQL()
    {

    }

    /*
    *  Open database connection
    *  Parameters:
    *  $db: database name
    *  $user: database user
    *  $pass: database password
    *  $host: database host address
    *  $port: database port
    */
    function connect($db, $user, $pass, $host="localhost",
$port=3306)
    {
      $this->_conn=mysql_connect($host.":".$port, $user, $pass)
        or die("mysql_connect() failed.");
      mysql_select_db($db, $this->_conn)
        or die("mysql_select_db() failed.");
      return true;
    }

    /*
    *  Open persistent database connection
    */
    function pconnect($db, $user, $pass, $host="localhost",
$port=3306)
```

```
    {
      $this->_conn=mysql_pconnect($host.":".$port, $user, $pass)
        or die("mysql_connect() failed.");
      mysql_select_db($db, $this->_conn)
        or die("mysql_select_db() failed.");
      return true;
    }

    /*
    *  Close database connection
    */
    function close()
    {
      return mysql_close($this->_conn);
    }

    /*
    *  Execute query
    *  Parameters:
    *  $qry: SQL query
    *  Returns: new result set object
    */
    function query($qry)
    {
      $rs=new Rs();
      if( !$rs->_rs=mysql_query($qry, $this->_conn) )
        return false;
      return $rs;
    }

    /*
    *  Returns HTML <select> tags
    *  Parameters:
    *  $qry: SQL query for finding values to use in <option> tags
    *  $name: name of the <select> tag
    *  $allowNull: is a null answer allowed?
    *  $selected: which id is selected, if any
    */
    function htmlSelect($qry, $name, $allowNull=false,
$selected=false)
    {
      $select="<select name=\"{$name}\">\n";
      $res=mysql_query($qry, $this->_conn);

      if($allowNull)
        $select.="<option></option>\n";

      while($arr=mysql_fetch_row($res) )
      {
        if($selected && $selected == $arr[0] )
          $select.="<option value=\"{$arr[0]}\"
selected>{$arr[1]}</option>\n";
        else
          $select.="<option
value=\"{$arr[0]}\">{$arr[1]}</option>\n";
      }
      $select.="</select>\n";

      return $select;
```

```
}

/*
*  Create SQL UPDATE string
*  Parameters:
*  $table: table to UPDATE
*  $arr: array of values to SET
*  $where: WHERE clause
*  $exclude: values in $arr to exclude from SET
*/
function updateStr($table, $arr, $where, $exclude="")
{
  $sql ="UPDATE {$table} SET ";

  $excl=explode(",", $exclude);

  $cols="";
  foreach($arr as $varname => $val)
    if(!in_array($varname, $excl))
      $cols.=$varname.",";
    else
      unset($arr[$varname]);

  $cols=substr($cols, 0, -1);

  $sql2="SELECT {$cols} FROM {$table}";

  $rs=new Rs();
  $rs->rs=mysql_query($sql2, $this->_conn);

  $types=$this->_fieldType($table, $cols);

  foreach($types as $colname=>$coltype)
  {
    // See if column type is numeric or text
    $sql.=$colname."=";
    if($coltype == "I" || $coltype == "N")
      if($arr[$colname] == "")
        $sql.="NULL,";
      else
        $sql.="{$arr[$colname]},";
    else
      if($arr[$colname] == "")
        $sql.="NULL,";
      else
        $sql.="'{$arr[$colname]}',";
  }

  $sql=substr($sql, 0, -1);

  $sql.=" WHERE ";

  $sql.=$where;

  return $sql;
}

/*
*  Find type of column
```

```
  *  Parameters:
  *  $table: Table in database
  *  $columns: comma separated list of column names
  *  Returns: array with colname/coltype pairs
  */
  function _fieldType($table, $columns)
  {
    $col=explode(",", $columns);

    $sql="SELECT ";
    foreach($col as $c)
      $sql.=$c.",";

    $sql=substr($sql, 0, -1);

    $sql.=" FROM {$table}";

    $arr=array();

    $res=mysql_query($sql, $this->_conn);

    $numfields=mysql_num_fields($res);

    for($i=0; $i < $numfields; $i++)
    {
      $type=mysql_field_type($res, $i);

      if($type == "int")
        $type="I";
      else if($type == "real")
        $type="N";
      else if($type == "date")
        $type="D";
      else if($type == "string")
        $type="C";

      $arr[ $col[$i] ] = $type;
    }

    return $arr;
  }

}

/*
* Resultset class. Holds the contents of a database query
*/
class Rs
{
  var $_numRows;
  var $_numFields;
  var $_rs;
  var $row=array();

  function Rs()
  {

  }
```

```
    /*
    *  Advance result set pointer to next row
    */
    function getNext($mode=DBL_BOTH)
    {
      if($mode == DBL_BOTH)
      {
        $this->row=mysql_fetch_array($this->_rs);
        if(!$this->row)
          return false;
        else
          return true;
      }
      else if($mode == DBL_ASSOC)
      {
        $this->row=mysql_fetch_array($this->_rs, MYSQL_ASSOC);
        if(!$this->row)
          return false;
        else
          return true;
      }
      else
      {
        $this->row=mysql_fetch_array($this->_rs, MYSQL_NUM);
        if(!$this->row)
          return false;
        else
          return true;
      }

    }

    /*
    *  Get number of rows in result set
    */
    function numRows()
    {
      $this->_numRows=mysql_num_rows($this->_rs);
      return $this->_numRows;
    }

    /*
    *  Get number of columns in result set
    */
    function numFields()
    {
      $this->_numFields=mysql_num_fields($this->_rs);
      return $this->_numFields;
    }

  } // Class Rs

?>
```

/inc/functions.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)
```

```
/*
*  Useful functions for the CMS system
*/

/*
*  Check if user is logged in
*  Returns: true if user is logged in, else false
*/
function isLoggedIn()
{
  if( @$_SESSION["userid"] > 0 )
    return true;

  return false;
}

/*
*  Check if user is an admin
*  Returns: true if user is admin, else false
*/
function isAdmin()
{
  if( @$_SESSION["usergroup"] == ADMINGROUP )
    return true;

  return false;
}

/*
*  Check permissions
*  Parameters:
*  $mod: module to check for
*  $access: bitmask of access permissions
*  Returns:  true if user has access, else false
*/
function hasAccess( $mod, $access )
{
  global $DBP,$db;

  if( isLoggedIn() )
  {
    // If user is an admin everything is allowed
    if( isAdmin() )
      return true;

    // Find module's ID
    $sql="SELECT id FROM {$DBP}Modules WHERE ";
    $sql.="modname='{$mod}'";

    $rs=$db->query($sql);
    $rs->getNext();

    // Search for ID or invalid one (0)
    isset($rs->row["id"]) ? $id=$rs->row["id"] : $id=0;

    // Build list of groups user is in
    $sql="SELECT groupid FROM {$DBP}GroupMembers ";
    $sql.="WHERE userid={$_SESSION["userid"]}";
```

```
      $rs=$db->query($sql);

      $grp=" AND (groupid={$_SESSION["usergroup"]}";

      while( $rs->getNext() )
        $grp.=" OR groupid=".$rs->row["groupid"]; //." OR ";

      $grp.=")";

      // Find permissions for user's group(s)
      $sql="SELECT perm ";
      $sql.="FROM {$DBP}ModPerms ";
      $sql.="WHERE module={$id} {$grp}";

      $rs=$db->query($sql);

      // Default to not having permission
      $hasAccess=false;

      while( $rs->getNext() )
      {
        // See if user has permissions
        if( ($rs->row["perm"] & $access) != 0)
          $hasAccess=true;
      }
      return $hasAccess;
    }
    return false;

  }
?>
```

/inc/loadlangfile.php

```
<?php
 // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

 // Attempt to load a language file for this module
 if( @!include(
"./modules/{$_GET["mod"]}/tpl/{$_COOKIE["userlang"]}/lang.{$_COOKI
E["userlang"]}.php" ) )
       if ( @!include(
"./modules/{$_GET["mod"]}/tpl/{$DEFAULT_LANGUAGE}/lang.{$DEFAULT_L
ANGUAGE}.php" ) )
             if ( @!include(
"./modules/{$_GET["mod"]}/tpl/lang.php" ) )
                   exit("Error. No language file was found.");
 // This should never happen
?>
```

/inc/globals.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

  // Default language
  $DEFAULT_LANGUAGE = "en";

  // Database table prefix
```

```
  $DBP = "";

  // Database settings
  $DBUSER = "cms";  // Username
  $DBPASS = "cms";  // Password
  $DBNAME = "cms";  // Database name
  $DBHOST = "localhost";  // Database host

  // Admin group's ID
  define("ADMINGROUP", 10);

  // Permissions for modules
  define("READ", 1);
  define("WRITE", 2);
  define("DELETE", 4);
  define("CREATE", 8);
  define("MODIFY", 16);

?>
```

/modules/admin/admin.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

  include dirname(__FILE__) . "/../../inc/loadlangfile.php";

  /*
  * Administration class
  */
  class Admin
  {
    function main()
    {
      global $MODLANG,$DBP,$db;

      $modResult="";

      // Must be admin to see anything
      if( !isAdmin() )
        return $modResult;

      switch( @$_GET["modop"] )
      {
        // Create new user
        case "newuser":
          $modTpl=new Template("./modules/Admin/tpl/status.tpl");

          // Does the user name exist already?
          $sql="SELECT id FROM {$DBP}Users WHERE
username='{$_POST["username"]}'";
          $mrs=$db->query($sql);
          if( $mrs->getNext() )
          {
            $modTpl->assign("!STATUS", "status_error");
            $modTpl->assign("!MESSAGE",
$MODLANG["STATUS_ERROR_USER_EXISTS"]);
          }
          else
```

```
          {
            // No, create the new user then
            $sql="INSERT INTO {$DBP}Users
VALUES(NULL,'{$_POST["username"]}',";

$sql.="'".md5($_POST["password"])."','{$_POST["fullname"]}',";

$sql.="'{$_POST["email"]}','{$_POST["address"]}','{$_POST["zip"]}'
,";

$sql.="'{$_POST["city"]}','{$_POST["country"]}','{$_POST["phone"]}
',";

$sql.="'{$_POST["mobilephone"]}',{$_POST["usergroup"]},";
            $sql.="{$_POST["userstatus"]},NOW(),NULL)";

            $modTpl->assign("!STATUS", "status_ok");
            $modTpl->assign("!MESSAGE",
$MODLANG["STATUS_USER_ADDED"]);

            $db->query($sql);
          }
          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // Show new user template
        case "users":
          $modTpl=new Template("./modules/Admin/tpl/users.tpl");
          $sql="SELECT id,groupname FROM {$DBP}Groups ORDER BY id
DESC";
          $primaryGroup=$db->htmlSelect($sql, "usergroup");
          $modTpl->assign("!PRIMARY_GROUP", $primaryGroup);

          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // Show edit user template
        case "edituser":
          $modTpl=new
Template("./modules/Admin/tpl/edituser.tpl");
          $sql="SELECT * FROM {$DBP}Users WHERE id={$_GET["id"]}";
          $mrs=$db->query($sql);

          $mrs->getNext();

          // Fill in this user's details
          $modTpl->assign("#!USERID", $_GET["id"]);
          $modTpl->assign("!USERNAME", $mrs->row["username"]);
          $modTpl->assign("!FULLNAME", $mrs->row["fullname"]);
          $modTpl->assign("!EMAIL", $mrs->row["email"]);
          $modTpl->assign("!ADDRESS", $mrs->row["address"]);
          $modTpl->assign("!ZIP", $mrs->row["zip"]);
          $modTpl->assign("!CITY", $mrs->row["city"]);
          $modTpl->assign("!COUNTRY", $mrs->row["country"]);
          $modTpl->assign("!PHONE", $mrs->row["phone"]);
          $modTpl->assign("!MOBILEPHONE", $mrs-
>row["mobilephone"]);
```

```
          $sql="SELECT id,groupname FROM {$DBP}Groups ORDER BY id
DESC";
          $primaryGroup=$db->htmlSelect($sql, "usergroup", false,
$mrs->row["usergroup"]);
          $modTpl->assign("!PRIMARY_GROUP", $primaryGroup);

          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // Show additional groups for this user
        case "usergroups":
          $modTpl=new
Template("./modules/Admin/tpl/usergroups.tpl");
          $modTpl->assign("!USERID", $_GET["userid"]);

          switch( @$_GET["func"] )
          {
            // Remove from group
            case "remove":
              $sql="DELETE FROM {$DBP}GroupMembers ";
              $sql.="WHERE groupid={$_GET["groupid"]} AND ";
              $sql.="userid={$_GET["userid"]}";

              $db->query($sql);
              break;

            // Add to group
            case "addgroup";
              $sql="INSERT INTO {$DBP}GroupMembers ";
              $sql.="VALUES
({$_GET["userid"]},{$_POST["groupid"]})";

              $db->query($sql);
              break;

            default:
              break;
          }

          // List groups that user belongs to
          $sql="SELECT userid,groupid,groupname ";
          $sql.="FROM {$DBP}GroupMembers ";
          $sql.="LEFT JOIN {$DBP}Groups ";
          $sql.="ON {$DBP}Groups.id={$DBP}GroupMembers.groupid ";
          $sql.="WHERE userid={$_GET["userid"]}";

          $mrs=$db->query($sql);

          while( $mrs->getNext() )
          {
            $modTpl->assign("!GROUPID", $mrs->row["groupid"]);
            $modTpl->assign("!GROUPNAME", $mrs->row["groupname"]);
            $modTpl->assign("!USERID", $_GET["userid"]);
          }

          $select=$db->htmlSelect("SELECT id,groupname FROM
{$DBP}Groups", "groupid");
```

```
          $modTpl->assign("!GROUPS", $select);
          $modTpl->assign("!USERID", $_GET["userid"]);
          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // Update user's data
        case "updateuser":
          $modTpl=new Template("./modules/Admin/tpl/status.tpl");
          if( @$_POST["passwd"] != "" )
          {
            $_POST["passwd"]=md5($_POST["passwd"]);
            $upd=$db->updateStr("{$DBP}Users", $_POST,
"id={$_POST["userid"]}", "userid");
          }
          else
            $upd=$db->updateStr("{$DBP}Users", $_POST,
"id={$_POST["userid"]}", "userid,passwd");

          // Check for success
          if( $db->query($upd) )
          {
            $modTpl->assign("!STATUS", "status_ok");
            $modTpl->assign("!MESSAGE",
$MODLANG["STATUS_USER_UPDATED"]);
          }
          else
          {
            $modTpl->assign("!STATUS", "status_error");
            $modTpl->assign("!MESSAGE",
$MODLANG["STATUS_ERROR_USER_NOT_UPDATED"]);
          }
          $modTpl->parse();
          $modResult=$modTpl->get();

          break;

        // List groups
        case "groups":
          $modTpl=new Template("./modules/Admin/tpl/groups.tpl");

          switch(@$_GET["func"])
          {
            // Update group's data
            case "updategroup":
              if( isset($_POST["f_update"]) )
              {
                $sql="UPDATE {$DBP}Groups SET
id={$_POST["groupid"]},";
                $sql.="groupname='{$_POST["groupname"]}' WHERE ";
                $sql.="id={$_POST["oldid"]}";
                $db->query($sql);
              }
              else if( isset($_POST["f_delete"]) )
              {
                $sql="DELETE FROM {$DBP}Groups WHERE
id={$_POST["oldid"]}";
                $db->query($sql);
```

```
                $sql="DELETE FROM {$DBP}ModPerms WHERE
groupid={$_POST["oldid"]}";
                $db->query($sql);
              }
              break;

            // Add new group
            case "addgroup":
              $sql="INSERT INTO {$DBP}Groups VALUES(";
              $sql.=$_POST["groupid"].",'{$_POST["groupname"]}')";

              $db->query($sql);
              break;

            default:
              break;
          }

          // List groups
          $sql="SELECT id,groupname FROM {$DBP}Groups WHERE
groupname!='Admin' ORDER BY id";
          $mrs=$db->query($sql);
          while($mrs->getNext() )
          {
            $modTpl->assign("!GROUPID", $mrs->row["id"]);
            $modTpl->assign("!OLDID", $mrs->row["id"]);
            $modTpl->assign("!GROUPNAME", $mrs->row["groupname"]);

          }
          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // List activated modules
        case "modules":
          switch( @$_GET["func"] )
          {
            // Inactive module
            case "inactivate":
              $sql="DELETE FROM {$DBP}Modules ";
              $sql.="WHERE id={$_GET["modid"]}";

              $db->query($sql);
              break;

            // Activate module
            case "activate":
              if($_POST["modname"] == "Admin")
                break;

              $sql="INSERT INTO {$DBP}Modules (modname) ";
              $sql.="VALUES ('{$_POST["modname"]}')";

              $db->query($sql);
              break;

            default:
              break;
          }
```

```
            // List all activated modules
            $modTpl=new Template("./modules/Admin/tpl/modules.tpl");

            $sql="SELECT id,modname FROM {$DBP}Modules WHERE
modname!='Admin'";
            $mrs=$db->query($sql);

            $mods=array();
            while($mrs->getNext() )
            {
              $modTpl->assign("!MODID", $mrs->row["id"]);
              $modTpl->assign("!MODNAME", $mrs->row["modname"]);
            }

            // Find all modules
            if($handle=opendir("./modules"))
            {
              while (false !== ($file = readdir($handle)))
              {
                if ($file != "." && $file != "..")
                {
                  if( is_dir("./modules/".$file) )
                  {
                    $modTpl->assign("!MODNAME2", $file);
                    $modTpl->assign("!MODNAME3", $file);
                  }
                }
              }
              closedir($handle);
            }

            $modTpl->parse();
            $modResult=$modTpl->get();
            break;

          // Show permissions
          case "permissions":
            switch( @$_GET["func"] )
            {
              case "new":
                $sql="INSERT INTO {$DBP}ModPerms ";
                $sql.="VALUES ({$_POST["module"]},";
                $sql.="{$_POST["groupid"]},";

                $p=0;
                if( isset($_POST["read"]) )
                  $p |= READ;
                if( isset($_POST["write"]) )
                  $p |= WRITE;
                if( isset($_POST["delete"]) )
                  $p |= DELETE;
                if( isset($_POST["create"]) )
                  $p |= CREATE;
                if( isset($_POST["modify"]) )
                  $p |= MODIFY;

                $sql.=$p.")";
```

```
                $db->query($sql);
              case "update":
              default:
                if( isset($_POST["f_update"]) )
                {
                  // Update permissions for module
                  $sql="UPDATE {$DBP}ModPerms SET perm=";

                  // Build bitmask
                  $p=0;
                  if( isset($_POST["read"]) )
                    $p |= READ;
                  if( isset($_POST["write"]) )
                    $p |= WRITE;
                  if( isset($_POST["delete"]) )
                    $p |= DELETE;
                  if( isset($_POST["create"]) )
                    $p |= CREATE;
                  if( isset($_POST["modify"]) )
                    $p |= MODIFY;

                  $sql.=$p." WHERE module={$_POST["module"]} ";
                  $sql.="AND groupid={$_POST["groupid"]}";

                  $db->query($sql);
                }
                // Delete permissions
                if( isset($_POST["f_delete"]) )
                {
                  $sql="DELETE FROM {$DBP}ModPerms ";
                  $sql.="WHERE module={$_POST["module"]} ";
                  $sql.="AND groupid={$_POST["groupid"]}";

                  $db->query($sql);
                }

                $modTpl=new
    Template("./modules/Admin/tpl/permissions.tpl");

                // List current permissions
                $sql="SELECT module,modname,groupid,groupname,perm
    ";
                $sql.="FROM {$DBP}ModPerms ";
                $sql.="LEFT JOIN {$DBP}Modules ";
                $sql.="ON {$DBP}Modules.id=module ";
                $sql.="LEFT JOIN {$DBP}Groups ";
                $sql.="ON {$DBP}Groups.id=groupid ";
                $sql.="WHERE modname!='Admin'";
                $mrs=$db->query($sql);

                while($mrs->getNext() )
                {
                  $modTpl->assign("!MODULE", $mrs->row["modname"]);
                  $modTpl->assign("!MODULEID", $mrs->row["module"]);
                  $modTpl->assign("!GROUP", $mrs->row["groupname"]);
                  $modTpl->assign("!GROUPID", $mrs->row["groupid"]);

                  if( $mrs->row["perm"] & READ )
                    $modTpl->assign("!READ", "checked=\"checked\"");
```

```
                else
                  $modTpl->assign("!READ", "");
                if( $mrs->row["perm"] & WRITE )
                  $modTpl->assign("!WRITE",
"checked=\"checked\"");
                else
                  $modTpl->assign("!WRITE", "");
                if( $mrs->row["perm"] & DELETE )
                  $modTpl->assign("!DELETE",
"checked=\"checked\"");
                else
                  $modTpl->assign("!DELETE", "");
                if( $mrs->row["perm"] & CREATE )
                  $modTpl->assign("!CREATE",
"checked=\"checked\"");
                else
                  $modTpl->assign("!CREATE", "");
                if( $mrs->row["perm"] & MODIFY )
                  $modTpl->assign("!MODIFY",
"checked=\"checked\"");
                else
                  $modTpl->assign("!MODIFY", "");


              }

              $modSelect=$db->htmlSelect("SELECT id,modname FROM
{$DBP}Modules", "module");
              $modTpl->assign("!MODULESELECT", $modSelect);
              $grpSelect=$db->htmlSelect("SELECT id,groupname FROM
{$DBP}Groups", "groupid");
              $modTpl->assign("!GROUPSELECT", $grpSelect);
              $modTpl->parse();
              $modResult=$modTpl->get();
              break;
          }
          break;
        // Display main Admin screen
        default:
          $modTpl=new Template("./modules/Admin/tpl/default.tpl");

          $sql="SELECT id,username,fullname,email FROM
{$DBP}Users";
          $mrs=$db->query($sql);

          while( $mrs->getNext() )
          {
            $modTpl->assign("!USERID", $mrs->row["id"]);
            $modTpl->assign("!USERNAME", $mrs->row["username"]);
            $modTpl->assign("!FULLNAME", $mrs->row["fullname"]);
            $modTpl->assign("!EMAIL", $mrs->row["email"]);
          }
          $modTpl->parse();
          $modResult=$modTpl->get();
          break;
      }

      return $modResult;
    }
```

```
  }
?>
```

/modules/article/article.php

```
<?php
  // (c) 2004 Jukka-Pekka Luukkonen (jpluukkonen AT gmail DOT com)

  include dirname(__FILE__) . "/../../inc/loadlangfile.php";

  define("ARTICLE_OK", 0);
  define("ARTICLE_LOCKED", 1);
  define("ARTICLE_INVISIBLE", 2);

  /*
  * Article class
  */
  class Article
  {

    function main()
    {
      global $MODLANG,$DBP,$db;

      $modResult="";

      switch( @$_GET["modop"] )
      {
        // Create new article
        case "newarticle":
          // Check permissions
          if( hasAccess( $_GET["mod"], WRITE ) )
          {
            $modTpl=new
Template("./modules/Article/tpl/newarticle.tpl");
            $modResult=$modTpl->get();
          }
          break;

        // Insert new article
        case "insertarticle":
          if( hasAccess( $_GET["mod"], WRITE ) )
          {
            $sql="INSERT INTO {$DBP}Articles
(author,articlestatus,createtime,articlename,content) ";
            $sql.="VALUES
({$_SESSION["userid"]},0,NOW(),'{$_POST["articlename"]}','{$_POST[
"content"]}')";

            $db->query($sql);

            $modTpl=new
Template("./modules/Article/tpl/result.tpl");
            $modTpl->assign("!RESULT", $MODLANG["ARTICLE_SAVED"]);
          }
          else
          {
            $modTpl=new
Template("./modules/Article/tpl/error.tpl");
```

```
            $modTpl->assign("!ERROR",
$MODLANG["ERR_ARTICLE_NOT_SAVED"]);
          }
          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // Show selected article
        case "showarticle":
          $article=$this->_parseArticle($_GET["id"]);

          if( !($article["status"] == ARTICLE_LOCKED ||
$article["status"]  == ARTICLE_INVISIBLE) ||
            hasAccess( $_GET["mod"], READ ) )
          {
            $modTpl=new
Template("./modules/Article/tpl/showarticle.tpl");
            $modTpl->assign("!AUTHOR", $article["author"]);
            $modTpl->assign("!TIME", $article["time"]);
            $modTpl->assign("!CONTENT", $article["content"]);
            if($article["lastedit"])
              $modTpl->assign("!LASTEDIT", $article["lastedit"]);
            else
              $modTpl->assign("!LASTEDIT", "-");

            if( hasAccess( $_GET["mod"], MODIFY ) || $this-
>_isAuthor($_GET["id"]) )
              $modTpl->assign("!EDIT", "<a
href=\"{#!PHP_SELF}?op=loadmod&amp;mod=Article&amp;modop=editartic
le&amp;artid=".$_GET["id"]."\">{$MODLANG["ARTICLE_EDIT"]}</a>");
            else
              $modTpl->assign("!EDIT", "");
          }
          else
          {
            // Can't show because of permissions
            $modTpl=new
Template("./modules/Article/tpl/status.tpl");
            $modTpl->assign("!STATUS", "status_error");
            $modTpl->assign("!MESSAGE",
$MODLANG["ERR_NO_PERMISSIONS"]);
          }

          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // Edit article
        case "editarticle":
          if( hasAccess( $_GET["mod"], MODIFY ) || $this-
>_isAuthor($_GET["artid"]) )
          {
            $sql="SELECT articlestatus FROM {$DBP}Articles ";
            $sql.="WHERE id={$_GET["artid"]}";
            $mrs=$db->query($sql);
            $mrs->getNext();

            if ($mrs->row["articlestatus"] == ARTICLE_LOCKED &&
```

```
              !( $this->_isAuthor($_GET["artid"]) || hasAccess(
$_GET["mod"], MODIFY|WRITE) ) )
            {
              $modTpl=new
Template("./modules/article/tpl/status.tpl");
              $modTpl->assign("!STATUS", "status_error");
              $modTpl->assign("!MESSAGE",
$MODLANG["ERR_ARTICLE_LOCKED"]);
            }
            else
            {
              // Has permissions to edit
              $sql="UPDATE {$DBP}Articles SET
articlestatus=".ARTICLE_LOCKED;
              $sql.="WHERE id={$_GET["artid"]}";
              $db->query($sql);

              $sql="SELECT articlename,content FROM {$DBP}Articles
";
              $sql.="WHERE id={$_GET["artid"]}";
              $mrs=$db->query($sql);
              $mrs->getNext();
              $modTpl=new
Template("./modules/article/tpl/editarticle.tpl");
              $modTpl->assign("!ARTID", $_GET["artid"]);
              $modTpl->assign("!ARTICLENAME", $mrs-
>row["articlename"]);
              $modTpl->assign("!ARTICLETEXT", $mrs-
>row["content"]);
            }
          }
          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // Insert updated article
        case "updatearticle":
          $modTpl=new
Template("./modules/article/tpl/status.tpl");

          if( hasAccess( $_GET["mod"], MODIFY )  || $this-
>_isAuthor($_POST["artid"]) )
          {
            $sql="UPDATE {$DBP}Articles SET
articlename='{$_POST["articlename"]}',";

$sql.="content='{$_POST["content"]}',articlestatus={$_POST["status
"]},";
            $sql.="lastedit=NOW() WHERE id={$_POST["artid"]}";

            $modTpl->assign("!STATUS", "status_ok");
            $modTpl->assign("!MESSAGE",
$MODLANG["ARTICLE_UPDATED"]);
            $db->query($sql);
          }
          else
          {
            // User didn't have permissions
            $modTpl->assign("!STATUS", "status_error");
```

```
            $modTpl->assign("!MESSAGE",
$MODLANG["ERR_ARTICLE_NOT_UPDATED"]);
          }
          $modTpl->parse();
          $modResult=$modTpl->get();
          break;

        // List articles
        default:
          $modTpl=new
Template("./modules/Article/tpl/default.tpl");

          $w="";
          // See which articles to list depending on permissions
          if( !hasAccess( $_GET["mod"], WRITE | MODIFY | CREATE )
)
            $w="AND articlestatus!=".ARTICLE_INVISIBLE;

          $sql="SELECT
{$DBP}Articles.id,fullname,articlename,createtime FROM
{$DBP}Articles,{$DBP}Users ";
          $sql.="WHERE author={$DBP}Users.id {$w} ORDER BY
createtime DESC";

          $mrs=$db->query($sql);

          while($mrs->getNext() )
          {
            $modTpl->assign("!ARTICLEID", $mrs->row["id"]);
            $modTpl->assign("!ARTICLENAME", $mrs-
>row["articlename"]);
            $modTpl->assign("!AUTHOR", $mrs->row["fullname"]);
            $modTpl->assign("!TIME", $mrs->row["createtime"]);
          }

          // Show additional menu if user has permissions
          if( hasAccess( $_GET["mod"], CREATE) )
          {
            $modTpl2=new
Template("./modules/article/tpl/topmenu.tpl");

            $modTpl->assign("!MENU", $modTpl2->get() );
          }
          else
            $modTpl->assign("!MENU", "");

          $modTpl->parse();
          $modResult=$modTpl->get();
          break;
      }

      return $modResult;
    }

    /*
    *  Check if the logged in user is author of article
    *  Parameters:
    *  $articleId: ID in database table
    */
```

```
    function _isAuthor( $articleId )
    {
      global $DBP,$db;
      $sql="SELECT author FROM {$DBP}Articles ";
      $sql.="WHERE id={$articleId}";

      $mrs=$db->query($sql);
      $mrs->getNext();

      if(@$_SESSION["userid"] == $mrs->row["author"])
        return true;

      return false;
    }

    /*
    *  Parse article for display
    *  Parameters:
    *  $articleId: ID in database table
    *  Returns: array with article's values
    */
    function _parseArticle( $articleId )
    {
      global $DBP,$db;

      $sql="SELECT
fullname,articlename,createtime,content,lastedit,articlestatus ";
      $sql.="FROM {$DBP}Articles ";
      $sql.="LEFT JOIN {$DBP}Users ";
      $sql.="ON {$DBP}Users.id={$DBP}Articles.author ";
      $sql.="WHERE {$DBP}Articles.id={$articleId}";

      $mrs=$db->query($sql);

      $mrs->getNext();

      $msg=$mrs->row["content"];
      $msg=trim(strip_tags($msg));

      if($msg!="")
      {
        $msg=" ".$msg;
        $msg=str_replace("\n","<br />",$msg);
        $msg=str_replace("[[","<1>",$msg);
        $msg=str_replace("]]","<2>",$msg);
        $msg=str_replace("[b]","<strong>",$msg);
        $msg=str_replace("[B]","<strong>",$msg);
        $msg=str_replace("[/b]","</strong>",$msg);
        $msg=str_replace("[/B]","</strong>",$msg);
        $msg=str_replace("[i]","<i>",$msg);
        $msg=str_replace("[I]","<i>",$msg);
        $msg=str_replace("[/i]","</i>",$msg);
        $msg=str_replace("[/I]","</i>",$msg);
        $msg=str_replace("[u]","<u>",$msg);
        $msg=str_replace("[U]","<u>",$msg);
        $msg=str_replace("[/u]","</u>",$msg);
        $msg=str_replace("[/U]","</u>",$msg);
        $msg=str_replace("[A","[a",$msg);
        $msg=str_replace("[\A","[\a",$msg);
```

```
        $msg=str_replace("[p]","<p>",$msg);
        $msg=str_replace("[/p]","</p>",$msg);
        $msg=str_replace("[h1]","<h1>",$msg);
        $msg=str_replace("[/h1]","</h1>",$msg);
        $msg=str_replace("[h2]","<h2>",$msg);
        $msg=str_replace("[/h2]","</h2>",$msg);
        $msg=str_replace("[h3]","<h3>",$msg);
        $msg=str_replace("[/h3]","</h3>",$msg);
        $msg=str_replace("[h4]","<h4>",$msg);
        $msg=str_replace("[/h4]","</h4>",$msg);
        $msg=str_replace("[h5]","<h5>",$msg);
        $msg=str_replace("[/h5]","</h5>",$msg);
        $msg=str_replace("[h6]","<h6>",$msg);
        $msg=str_replace("[/h6]","</h6>",$msg);


        while(preg_match("/\[a (\S*)\](.*)\[\/a\]/i",$msg,$arr))
          $msg=str_replace($arr[0],"<a
href=\"".$arr[1]."\">".$arr[2]."</a>",$msg);

        while(preg_match("/\[img (\S*)\]/i",$msg,$arr))
          $msg=str_replace($arr[0],"<img src=\"".$arr[1]."\"
border=\"0\">",$msg);

        $msg=str_replace("<1>","[",$msg);
        $msg=str_replace("<2>","]",$msg);
      }

      $arr=array("author"=>$mrs->row["fullname"],
        "time"=>$mrs->row["createtime"],
        "content"=>$msg,
        "lastedit"=>$mrs->row["lastedit"],
        "status"=>$mrs->row["articlestatus"]);

      return $arr;
    }
  }
?>
```