

A Content Management System (CMS) is used to add, edit, and delete content on a website. For a small website, such as this, adding and deleting a page manually is fairly simple. But for a large website with lots of pages like a news website adding a page manually without a content management system can be a headache.

A CMS is meant to ease the process of adding and modifying new content to a webpage. The pages content are stored in database, not in the file server.

This tutorial will present an example of a simple content management system. You will be able to add, edit and delete articles using HTML forms.

For the database table we'll call it the news table. It consist of three columns :

- id : The article's id
- title : The title of an article
- content : The article itself

First we need to create a script to add an article. It is just a form where a user can enter the article's title and content.

Example :

```
<form method="post">
<table width="700" border="0" cellpadding="2" cellspacing="1" align="center">
<tr>
<td width="100">Title</td>
<td><input name="title" type="text"></td>
</tr>
<tr>
<td width="100">Content</td>
<td><textarea name="content" cols="50" rows="10"></textarea></td>
</tr>
<tr>
<td width="100">&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td colspan="2" align="center"><input name="save" type="submit" value="Save
Article"></td>
</tr>
</table>
</form>
```

When an article is added the script just insert the article into the database. An article id is automatically generated by MySQL because the id column was created with AUTO_INCREMENT parameter .

```
<?php
if(isset($_POST['save']))
```

```
{
    $title = $_POST['title'];
    $content = $_POST['content'];

    if(!get_magic_quotes_gpc())
    {
        $title = addslashes($title);
        $content = addslashes($content);
    }
    include 'library/config.php';
    include 'library/opendb.php';

    $query = " INSERT INTO news (title, content) ".
        " VALUES ('$title', '$content')";
    mysql_query($query) or die('Error ,query failed');

    include 'library/closedb.php';

    echo "Article '$title' added";
}
?>
```

Now that we have the script to add articles let's create another script to view those articles. The script is list the title of articles available in database as clickable links. The article link have the article id appended like this

<http://www.php-mysql-tutorial.com/examples/cms/article1.php?id=3>

One possible implementation of article1.php is presented below :

Example :

```
<?php
include 'library/config.php';
include 'library/opendb.php';

// if no id is specified, list the available articles
if(!isset($_GET['id']))
{
    $self = $_SERVER['PHP_SELF'];

    $query = "SELECT id, title FROM news ORDER BY id";
    $result = mysql_query($query) or die('Error : ' . mysql_error());

    // create the article list
    $content = '<ol>';
    while($row = mysql_fetch_array($result, MYSQL_NUM))
    {
```

```
list($id, $title) = $row;
$content .= "<li><a href=\"\$self?id=$id\">$title</a></li>\r\n";
}

$content .= '</ol>';

$title = 'Available Articles';
} else {
    // get the article info from database
    $query = "SELECT title, content FROM news WHERE id=".$_GET['id'];
    $result = mysql_query($query) or die('Error : ' . mysql_error());
    $row = mysql_fetch_array($result, MYSQL_ASSOC);

    $title = $row['title'];
    $content = $row['content'];
}

include 'library/closedb.php';
?>

// ... more code here
```

When article1.php is first called the \$_GET['id'] variable is not set and so it will query the database for the article list and save the list in the \$content variable as an ordered list. The variable \$title and \$content will be used later when we print the result page. Take a look at the code below :

Example :

```
<?php
```

```
// ... previous code
```

```
?>
<html>
<head>
<title>
<?php echo $title; ?>
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
```

```
// ... some css here to make the page look nicer
```

```
</style>
</head>
<body>
<table width="600" border="0" align="center" cellpadding="10" cellspacing="1"
bgcolor="#336699">
<tr>
<td bgcolor="#FFFFFF">
```

```
<h1 align="center"><?php echo $title; ?></h1>
<?php
echo $content;

// when displaying an article show a link
// to see the article list
if(isset($_GET['id']))
{
?>
<p>&nbsp;</p>
<p align="center"><a href="<?php echo $_SERVER['PHP_SELF']; ?>">Article
List</a></p>
<?php
}
?>
</td>
</tr>
</table>
</body>
</html>
```

If you click on an article link the script will fetch the article's title and content from the database, save it to \$title and \$content variable and print the HTML file . At the bottom of the page we place a code to show the link to the article list which is the file itself without any query string (\$_SERVER['PHP_SELF'])

With this implementation each article request involve one database query. For a heavy load website with lots of articles using the above implementation can cause a very high amount of database-request. So we need a better cms solution to reduce the load.

One feasible solution is to implement caching (cache) which load an article from the database only once when the article was first requested. The article is then saved to a cache directory as a regular HTML file. Subsequent request to the article will no longer involve any database request. The script just need to read the requested article from the cache directory.

Example :

```
<?php
include 'library/config.php';
include 'library/opendb.php';

$cacheDir = dirname(__FILE__) . '/cache/';

if (isset($_GET['id'])) {
    $cacheFile = $cacheDir . '_' . $_GET['id'] . '.html';
} else {
    $cacheFile = $cacheDir . 'index.html';
}

if (file_exists($cacheFile))
```

```
/http://www.php-mysql-tutorial.com
```

```
{  
    header("Content-Type: text/html");  
    readfile($cacheFile);  
    exit;  
}
```

```
// ... more code coming
```

```
?>
```

First we need to specify the cache directory where all cache files are located. For this example the cache directory is located in the same place as the article2.php script. I mean if article2.php is stored in C:/webroot then the cache dir is in C:/webroot/cache/

The script then check if the article was already in the cache. An article is saved into the cache directory using a filename generated from its id. For example if you request the article using a link like this :

<http://www.php-mysql-tutorial.com/examples/cms/article2.php?id=3>

Then the cache file for the article is

_3.html

This filename is just an underscore (_) followed by the article id. In case article2.php is called like this :

<http://www.php-mysql-tutorial.com/examples/cms/article2.php>

no id is defined so we make the cache file name as index.html

If the cache file is found , the content is read and printed using readfile() and the script terminate. When the article is not found in the cache then we need to look in the database and get the page content from there.

Example :

```
<?php
```

```
// ... previous code
```

```
if(!isset($_GET['id']))  
{  
    $self = $_SERVER['PHP_SELF'];  
    $query = "SELECT id, title FROM news ORDER BY id";  
    $result = mysql_query($query) or die('Error : ' . mysql_error());
```

```
$content = '<ol>';  
while($row = mysql_fetch_array($result, MYSQL_NUM))  
{  
    list($id, $title) = $row;  
    $content .= "<li><a href=\"\$self?id=$id\">$title</a></li>\r\n";  
}
```

```
$content .= '</ol>';
```

```
$title = 'Available Articles';
} else {
// get the article info from database
$query = "SELECT title, content FROM news WHERE id=".$_GET['id'];
$result = mysql_query($query) or die('Error : ' . mysql_error());
$row = mysql_fetch_array($result, MYSQL_ASSOC);

$title = $row['title'];
$content = $row['content'];
}

include 'library/closedb.php';
// ... still more code coming
?>
```

As you can see above the process of fetching the article list and content is the same as article1.php. But before showing the page we have to start output buffering so we can save the content of the generated HTML file.

See the code below. Just before printing the html we call `ob_start()` to activate output buffering. From this point no output is sent from the script to the browser. So in the code example below anything between `<html>` and `</html>` tag is not sent to the browser but stored in an internal buffer first.

After the closing html tag we use `ob_get_contents()` to get the buffer content and store it in a temporary variable, `$buffer`. We then call `ob_end_flush()` which stops the output buffering (so the page is now sent to the browser).

Example :

```
<?php
// ... previous code
ob_start();
?>
<html>

// ... same html code as article1.php
</html>
<?php
// get the buffer
$buffer = ob_get_contents();
// end output buffering, the buffer content
// is sent to the client
ob_end_flush();

// now we create the cache file
$fp = fopen($cacheFile, "w");
fwrite($fp, $buffer);
fclose($fp);
?>
```

Now that we have the file content we can write the cache file using the filename generated earlier (using underscore plus the article id). From now on any request to the article will no longer involve a database query. At least until the article is updated.