**BUILD A CMS**

**PHP – example**

**Overview**

Functionality:

* Pages can be added
* The most recent pages are previewed on the home page
* An individual page can be seen in its entirety
* A user can log in
* A user can logout
* Only administrative users, or the author of the currently viewed page of content, will be provided access to the edit page
* Only administrative users or author users can create new pages

This can easily be extended in terms of functionality.

The example will use a light, makeshift Model-View-Controller (MVC). MVC is a common software architectural pattern. Using MVC, you separate the data (i.e. the model), from the output (i.e. the view), using the controller as the agent.

In a web environment, the models are normally represented by database tables, although some models can also represent form data that doesn’t get stored in the database (such as that use in a contact form). Naturally, the views on the webpage are the HTML pages – the final, dynamically generated output that the user actually sees. The controllers react to user actions, such as the request of a single page, or the submission of a form. Controllers implement the logic: validate some data, insert it into a database, show the results and so forth.

By using MVC, you’ll find that your project will be easier to expand & easier to maintain. You can add and change any of the 3 components without necessarily touching the others.

The models will always be classes. The view files are primarily HTML, with very little logic (bare minimum of PHP code). In more formal MVC structures, you would use objects as controllers. In this example, individual PHP scripts will act as controllers.

Directory:

**Htdocs**

Classes

CSS

Images

Includes

JS

Index.php

Page.php

**Creating the database**

Always begin with the underlying database when it comes to web development. There will be 2 tables: users & pages.

|  |  |
| --- | --- |
| **Users** | |
| **Column** | **Type** |
| Id | INT |
| userType | ENUM |
| Username | VARCHAR(30) |
| Email | VARCHAR(40) |
| Pass | CHAR(40) |

|  |  |
| --- | --- |
| **Pages** | |
| **Column** | **Type** |
| Id | INT |
| creatorID | INT |
| Title | VARCHAR(100) |
| Content | TEXT |
| dataUpdated | TIMESTAMP |
| dateAdded | TIMESTAMP |

The id column from the users table is a foreign key in the pages table, indicating who created the page.

*\* if you want the ability to associate categories or tags with pages or posts, you would need to create a categories or tags table. Then you would need to create a pages\_categories (or pages\_tags or posts\_categories or posts\_tags) table that would act as the intermediary for the many-to-many relationships between pages/posts and categories/tags \**

*\* if you want users to be able to add comments to a page (or post), create a comments table. It would need to store the page ID, the comment content, the date submitted, and some reference to the user. What that reference would be would depend on whether or not only registered users could comment \**

Example

CREATE DATABASE cms;

USE cms;

CREATE TABLE users (

Id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

userType ENUM(‘public’, ‘author’, ‘admin’),

username VARCHAR(30) NOT NULL,

pass CHAR(40) NOT NULL,

dataAdded TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (id)

UNIQUE (username),

UNIQUE (email),

INDEX login (email, pass)

);

CREATE TABLE pages (

Id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

creatorId INT UNSIGNED NOT NULL,

title VARCHAR(100) NOT NULL,

content TEXT NOT NULL,

dateUpdated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

dataAdded TIMESTAMP NOT NULL,

PRIMARY KEY (id),

INDEX (creatorId),

INDEX (dateUpdated)

);

//now insert some data in to the tables

**Making the template**

This entire site can use a three-include approach:

* Include(‘includes/header.inc.php’);
* Include(‘views/specific\_content.html’);
* Include(‘includes/footer.inc.php’);

Create a standard, basic header & footer, use conditionals in the navbar for the header to display *if user logged in, show logout btn, else show login btn.* This can also be included in the footer. Also, in the footer, looking at the sitemap, include a conditional which state’s; *if user logged in, and user is allowed to create a page, link to ‘add\_page.php’.* (this example can be found on p291-293.

**Writing a utilities file**

This file will be included by every primary PHP script (i.e. every controller). On other sites, this might be called a *configuration* file, but this page does a bit more:

1. It defines the class loading function
2. It starts the session and checks for the presence of a user object previously stored in the session
3. It opens the database connection, creating a PDO object in the process

It is very easy to store simple data types, i.e. strings and numbers, in a session, file or database. But complex data types such as arrays & objects, are not easily stored in their original state. The solution is to convert the complex data type into a simple data type; using *serialize()*:

$data = array(‘Ali’ => ‘Hull’, ‘Joe’ => ‘Manchester’, ‘Shaida’ => ‘Paris’);

$sData = serialize($data);

The *serialize()* function outputs a string that represents the complex data.

From that string, the data can be reconstituted into its complex format via the *unserialize()* function:

$data = unserialize($sData);

If you store an object in the session, PHP will automatically serialize and unserialize the data on the fly, only if PHP can access the corresponding class definitions when the session is started again.

Example

//auto load classes from “classes” directory:

Function class\_loader($class) {

Require(‘classes/’ .$class . ‘.php’);

}

Spl\_autoload\_register(‘class\_loader’);

Session\_start(); //start the session

//check for a user in the session:

$user = (isset($\_SESSION[‘user’])) ? $\_SESSION[‘user’]: null;

//create the database connection as a PDO object

Try {

$pdo = new PDO(‘mysql:dbname=cms; host=localhost’, ‘username’, ‘password’);

}catch (PDOException $e) {

$pageTitle = “Error”; //pageTitle echoed in header.inc.php

Include(‘includes/header.inc.php’);

Include(‘views/error.html’);

Include(‘includes/footer.inc.php’);

Exit(); //IMPORTANT!!

}

**Creating the error view file**

<section>

<article>

<h2>An Error Occurred!</h2>

<p>The content is not viewable because an error occurred. We apologise for any inconvenience</p>

//not for public to see

<p>Details: <?php echo $e->getMessage(); ?></p>

</article>

</section>

**Defining the classes**

This CMS uses 2 user-defined classes: Page & User.

The Page class will be used to represent a page of content. This will come up in several places:

* Homepage – which shows the 3 most recent additions
* On an individual page – which displays an entire page of content
* When new pages of content are added

Class Page {

//all attributes correspond to database columns

Protected $id = null;

Protected $creatorId = null;

Protected $title = null;

Protected $content = null;

Protected $dataAdded = null;

Protected $dateUpdated = null;

//’getter’ methods to return attributes

Function getId() {

Return $this->id;

}

//do same for all other attributes

//returns the first x characters from the content

Function getIntro($count = 200) {

Return substr(strip\_tags($this->content), 0, $count) . ‘…’;

}

}//end of Page class

The purpose of the getIntro() method is to return an abbreviated, initial part of the page’s content as a way of previewing the page.

The User class represents registered & logged in users only (i.e. users not logged in are not represented by this class).

Class User {

//attributes match the corresponding DB columns

Protected $id = null;

Protected $userType = null;

Protected $username = null;

Protected $email = null;

Protected $pass = null;

Protected $dataAdded = null;

//return user id only

Function getId() {

Return $this->id;

}

//the following methods return a Boolean

//if user an administrator

Function isAdmin() {

Return ($this->userType == ‘admin’);

}

//if administrator or the user is the original author of provided page

Function canEditPage(Page $p) {

Return ($this->isAdmin() || ($this->id == $page->getCreatorId()));

}

//if user is administrator or author

Function canCreatePage() {

Return ($this->isAdmin() || ($this->userType == ‘author’));

}

} //end of User class

**Creating the homepage**

The homepage will retrieve the 3 most recent pages of content, display previews of them, and link to the script where the viewer can see the entire content. Along with this PHP script, you’ll need to create the view file that’s included by this page.

\* note differences between *include* and *require*. If script fails, *included* files will continue without the included content, whereas *required* will terminate the script \*

Example

//index.php

//include the utilities file

//set the page title and include the header.inc file

//fetch the 3 most recent pages:

Try {

$q = ‘SELECT id, title, content, DATE\_FORMAT(dateAdded, “%e %M %Y”) AS dateAdded FROM pages ORDER BY dateAdded DESC LIMIT 3’;

$r = $pdo->query($q);

//check that rows were returned

If($r &&& $r->rowCount() > 0) {

//set the fetch mode

$r->setFetchMode(PDO::FETCH\_CLASS, ‘Page’);

//Records will be fetched in the view:

Include(‘views/index.html’);

}else{ //problem

Throw new Exception(‘No content is available to be viewed at this time.’);

}

} catch (Exception $e) {

Includes(‘views/error.html’);

}

//include the footer

//index.html

<section>

<?php //fetch results and display

While($page = $r->fetch()) {

Echo “<article>

<h1><span>{$page->getDateAdded()}</span>{$page->getTitle()}</h1>

<p>{$page->getIntro()}</p>

<p><a href=\”page.php?id={$page->getId()}\”>Read more here…</a></p>

</article>”;

}

?>

</section>

**Viewing a page**

The homepage shows previews of the 3 most recently added pages of content. Each is linked to page.php, passing along the page’s ID value in the URL.

The purpose of page.php is to display the full content. This is a simple process:

* Validate the page ID
* Retrieve the corresponding database record
* Include the view file

Example

//page.php

//require the utilities file

Try {

//validate page ID

If(!isset($\_GET[‘id’]) || !filter\_var($\_GET[‘id’], FILTER\_VALIDATE\_INT, array(‘min\_range’ => 1))) {

Throw new Exception(‘An invalid page ID was provided to this page.’);

}

//fetch the page from the database:

$q = ‘SELECT id, title, content, DATE\_FORMAT(dateAdded, “%e %M %Y”) AS dateAdded FROM pages WHERE id=:id’;

$stmt = $pdo->prepare($q);

$r = $stmt->execute(array(‘:id’ => $\_GET[‘id’]));

//if the query ran ok, fetch the record in to an object

If($r) {

$stmt->setFetchMode(PDO::FETCH\_CLASS, ‘Page’);

$page = $stmt->fetch();

//confirm that it exists

If($page) {

//set the browser title to the page title

$pageTitle = $page->getTitle();

//create the page

Include(‘includes/header.inc.php’);

Include(‘views/page.html’);

}else{

Throw new Exception(‘An invalid page ID was provided to this page’);

}

}else{

Throw new Exception(‘An invalid page ID was provided to this page.’);

}

} catch (Exception as $e) { //catch generic exceptions

$pageTitle = “Error!”;

Include(‘includes/header.inc.php’);

Include(‘views/error.html’);

}

//include the footer

//page.html

<section>

<article>

<h1><span><?php echo $page->getDateAdded(); ?></span><?php echo $page->getTitle(); ?></h1>

<?php if($user && $user->canEditPage($page)) {

Echo ‘<p><a href=”edit\_page.php?id=’ . $page->getId() . ‘”>EDIT</a></p>’;

} ?>

</article>

</section>

**Using HTML\_QUICKFORM2**

HTML\_QUICKFORM2 is a PEAR ([www.pear.php.net](http://www.pear.php.net)) class that handles:

* Different types of form elements
* Different ways to validate data (numbers vs emails etc…)
* Displaying of errors
* And more…

This site has 2 forms: Login, and the creation of new pages of content.

Note that to use HTML\_QUICKFORM2, you must install it through the command line: *pear install HTML\_QUICKFORM2*

You must then include it in the scripts you will use it in.

To create form:

$form = new HTML\_QUICKFORM2(‘form\_id’);

$elem = $form->addElement(‘type’, ‘name’, ‘attr’, ‘data’);

//once all elements created

Echo $form; display form

To filter and validate form:

$elem->addFilter(‘function\_name’);

$elem->addFilter(‘trim’);

//to add rules

$elem->addrule(‘rule type’, ‘error message’);

$elem->addRule(‘required’, ‘Enter a value’);

//see rules on p314, table 9.5

To process form data

//if form passes validation

If($form->validate()){//code}

To access the form values:

$elem->getValue();

Example

//login.php

//require the utilities file

//create a new form

Set\_include\_path(get\_include\_path() . PATH\_SEPERATOR . ‘/usr/local/pear/share/pear/’);

Require(‘HTML/QuickForm2.php’);

$form = new HTML\_QuickForm2(‘loginForm’);

//email

$email = $form->addElement(‘text’, ‘email’);

$email->setLabel(‘Email Address’);

$email->addFilter(‘trim’);

$email->addRule(‘required’, ‘Please enter your email address’);

$email->addRule(‘email’, ‘Please enter your email address’);

//password

$password = $form->addElement(‘password’, ‘pass’);

$password->setLabel(‘Password’);

$password->addFilter(‘trim’);

$password->addRule(‘required’, ’Please enter your password’);

//submit btn

$form->addElement(‘submit’, ‘submit’, array(‘value’ => ‘login’));

//check for a form submission

If($\_SERVER[‘REQUEST\_METHOD’] == ‘POST’) { //handle the form

//validate form data

If($form->validate()) {

//check against the database

$q = ‘SELECT id, userType, username, email FROM users WHERE email=:email AND pass=SHA1(:pass)’;

$stmt = $pdo->prepare($q);

$r = $stmt->execute(array(‘:email’ => $email->getValue(), ‘:pass’ => $password->getValue()));

//try to fetch results

If($r) {

$stmt->setFetchMode(PDO::FETCH\_CLASS, ‘User’);

$user = $stmt->fetch();

}

//store the user in the session and redirect:

If($user) {

//store in a session

$\_SESSION[‘user’] = $user;

//redirect

Header(‘location:index.php’);

Exit;

}

} // end of form validation IF

}//end of form submission IF

//show the login page

$pageTitle = “Login”;

Include(‘includes/header.inc.php’);

Include(‘views/login.html’);

Include(‘includes/footer.inc.php’);

//login.html

<section>

<article>

<h1>Login</h1>

<?php if ($form->isSubmitted() && $form->validate()) {

Echo ‘<p>The values submitted do not match those on file!</p>’;

}?>

<?php echo $form; ?>

</article>

</section>

//the conditional above (login.html) checks if the form was submitted and passed validation. If so, the only reason the form is being shown again is because the provided values didn’t match those in the database

**Logging out**

1. Clear the session data in the array
2. Clear the session cookie
3. Destroy the session

Example

//logut.php

//require utilities file

//check for user before attempting to actually log them out:

If($user) {

$user = null;

$\_SESSION = array();

Setcookie(session\_name(), false, time()-3600);

Session\_destroy();

}

$pageTitle = “Logout”;

Include(‘includes/header.inc.php’);

Include(‘views/logout.html’); //need the view

Include(‘includes/footer.inc.php’);

//logout.html

<section><article><h1>You are now logged out</h1></article></section>

**Adding pages**

The most important feature of the CMS: the ability to dynamically add content

Example

//add\_page.php

//require the utilities file

//redirect user if the user doesn’t have permission:

If(!$user->canCreatePage()) {

Header(‘location: index.php’);

Exit;

}

//create a new form:

Set\_include\_path(get\_include\_path() . PATH\_SEPERATOR . ‘/usr/local/pear/share/pear/’);

Require(‘HTML/QuickForm2.php’);

$form = new HTML\_QuickForm2(‘addPageForm’);

//add the title field:

$title = $form->addElement(‘text’, ‘title’);

$title->setLabel(‘Page Title’);

$title->addFilter(‘strip\_tags’);

$title->addRule(‘required’, ‘Please enter a page title’);

//add the content field:

$content = $form->addElement(‘textarea’, ‘content’);

$content->setLabel(‘Page Content’);

$content->addFilter(‘trim’);

$content->addRule(‘required’, ‘Please enter the page content’);

//add the submit btn

$submit = $form->addElement(‘submit’, ‘submit’, array(‘value’=>’Add This Page’));

//check for a form submission

If($\_SERVER[‘REQUEST\_METHOD’] == ‘POST’) { //handle the form submission

//validate form data

If($form->validate()) {

//insert into the database:

$q = ‘INSERT INTO pages (creatorId, title, content, dateAdded) VALUES (:creatorId, :title, :content, NOW());

$stmt = $pdo->prepare($q);

$r = $stmt->execute(array(‘:creatorId’ => $user->getId(), ‘:title’ => $user- >getValue(), ‘:content’ => $user->getValue()));

//freeze the form upon success

If($r) {

$form->toggleFrozen(true); //makes form inputs un-editable

$form->removeChild($submit); //removes submit btn from form

}

} //end of form validation IF

}//end of form submission IF

//show page

$pageTitle = “Add a Page”;

Include(‘includes/header.inc.php’);

Include(‘views/add\_page.html’);

Include(‘includes/footer.inc.php’);

//add\_page.html

<section>

<article>

<h1>Add a new page of content</h1>

<?php if($form->isSubmitted() && $form->validate()) {

Echo ‘<p>The page has been added!</p>’;

}?>

<?php echo $form; ?>

</article>

</section>