**Modularising a website**

**Modularising a website**

Creating the config file

Every web app should begin with a configuration file. It serves several purposes, the main being:

* Defining constants
* Establishing site wide settings
* Creating user functions
* Managing errors

Basically, any piece of information that every page in a site might need to access.

\*side note – if a function would not likely be used by the majority of the site pages, put it in a separate file\*

<?php

**//settings**

$contact\_email = ‘address@example.com’; //errors to be emailed here

//determine whether we are working on a local server or real server

$host = substr($\_SERVER[‘HTTP\_HOST’], 0, 5);

If(in\_array($host, array(‘local’, ‘127.0’, ‘192.1’))) {

$local = true;

}else{

$local = false;

}

//determine location of files and the URL of the site:

If($local) {

//always debug when running locally

$debug = true;

//define the constants

Define(‘BASE\_URI’, ‘/path/to/html/folder’);

Define(‘BASE\_URL’, ‘http://localhost/directory/’);

Define(‘DB’, ‘/path/to/mysql.inc.php’);

}else{

Define(‘BASE\_URI’, ‘/path/to/live/html/folder’);

Define(‘BASE\_URL’, ‘http://example.com’);

Define(‘DB’, ‘/path/to/live/mysql.inc.php’);

}

/\*\*

\* to debug a specific page, add this to the index.php page:

\* if($p === ‘thismodule’) $debug = true; //before requiring this script

\* to debug the entire site, type the following code here (in config file):

\* $debug = true;

\*/

If(!isset($debug)) {

$debug = false;

}

**//error management**

//create the error handler

Function my\_error\_handler($e\_number, $e\_message, $e\_file, $e\_line, $e\_vars) {

Global $debug, $contact\_email;

//build the error message

$message = “An error occurred in script ‘$e\_file’ on line $e\_line: $e\_message”;

//append $e\_vars to the message:

$message .= print\_r($e\_vars, 1);

If($debug) { //show the error

Echo ‘<div class=”error”>’ . $message . ‘</div>’;

Debug\_print\_backtrace();

} else {

//log the error

Error\_log($message, 1, $contact\_email); //send email

//only print an error message if the error isn’t a notice or strict.

If( ($e\_number != E\_NOTICE) && ($e\_number < 2048) ) {

Echo ‘<div class=”error”>A system error occurred. We apologise for the inconvenience.</div>’;

}

}

}

//use my error handler

Set\_error\_handler(‘my\_error\_handler’);

?>

**Creating the HTML template file**

* Header.html
* Footer.html

In the navigation (in header.inc.html), add a $p to the url for all pages. E.g.:

<a href=”index.php”>Home</a>

<a href=”index.php?p=about”>About</a>

//note that all pages should lead to the index.php page, this is the only page sent to the browser

**Creating the index page**

The index page is the main script in the modularised application. In fact, it is the only page that should ever be loaded in the web browser. The technical term for this construct is *bootstrap file*, and it is also the common approach in framework-based sites.

The main purpose of the index page is to assemble all the proper pieces to complete the webpage:

* Include the configuration file
* Include the DB connectivity file
* Incorporating a HTML template
* Determining and including the proper content module

Example:

<?php

Require(‘./includes/config.inc.php);

//validate what page to show:

If(isset($\_GET[‘p’])) {

$p = $\_GET[‘p’]; //passed through URL

}else if(isset($\_POST[‘p’])) { //forms

$p = $\_POST[‘p’]; //passed through hidden input

}else{

$p = NULL;

}

//determine what page to display

Switch($p) {

Case ‘about’:

$page = ‘about.inc.php’;

$page\_title = ‘ABOUT’;

Break;

Case ‘contact’:

$page = ‘contact.inc.php’;

$page\_title = ‘CONTACT’;

Break;

Default:

$page = ‘main.inc.php’;

$page\_title = ‘HOMEPAGE’;

Break;

}

//make sure files exist

If(!file\_exists(‘./modules/’ . $page)) {

$page = ‘main.inc.php’;

$page\_title = ‘HOMEPAGE’;

}

Include(‘./includes/header.html’);

Include(‘./modules/’ . $page);

Include(‘./includes/footer.html’);

?>

**Creating content modules**

<?php

//redirect if page accessed directly

If(!defined(‘BASE\_URI’)) { //defined in config file

Require(‘../includes/config.inc.php’);

//redirect

$url = BASE\_URL . ‘index.php’;

Header(“location: $url”);

Exit;

}

?>

//HTML content below

You can also check for other things in the *if* statement to check if page accessed directly, e.g. check the $\_GET[‘p’] is set and is valid.

**Improved SEO with mod rewrite**

The primary Apache config file is httpd.conf. On shared hosts you cannot edit this direct so you may need to create a .htaccess file. Editing httpd.conf is preferred if available.

An .htaccess file is just a plain text file. The initial period (.) makes this a hidden file on the system.

To allow .htaccess override:

Httpd.conf

<Directory “path/to/directory”>

AllowOverride All

</Directory>

//save and restart apache

Enabling URL rewriting

Apache, via URL rewriting, takes the more user-friendly URL and parses it into something useable by the PHP scripts. Regular expressions are needed to most flexibly find matches. For example; to get *www.example.com/category/23* from *www.example.com/category.php?id=23:*

RewriteRule ^category/([0-9]+)/?$

Category.php?id=$1

Example:

<IfModule mod\_rewrite.c>

RewriteEngine on //turn on engine

RewriteBase /ch02/ //e.g. http://localhost/ch02/

RewriteRule ^(about|contact|this|that)/?$ index.php?p=$1

</IfModule>