

# Developers Guide

Software Version 9.0

## Table of contents

Table of contents .....	2
Introduction .....	3
Who are you? .....	3
Tools you may require .....	3
Getting Started.....	4
The eMeeting Framework .....	5
File Structure and Locations .....	6
How the system works.....	9
How are the pages determined?.....	10
How do I add a new page? .....	11
Adding and editing page content.....	17
Creating Sub pages .....	19
Posting Form Data.....	20
How is data send and received by the system?.....	21
Handeling Submitted Data .....	23
MYSQL Structure.....	25
Making Database Queries .....	26
Data Dictionary .....	26
Retrieving a single row of data.....	27
Retrieving More than one value .....	28
The Config file.....	30
Language Files .....	31

## Introduction

This guide has been put together to enable those who wish to customise and extend the software functionality the ability to do so by starting in the right place.

The document covers everything you would need to know to further develop and create a customised website application using the eMeeting framework.

### Who are you?

To read and understand this documentation I would recommend you have a good understanding of HTML and a basic understanding of PHP. The elements we will be discussing will be mostly of file and data structure therefore you don't have to be an expert in PHP to understand the code, but simply understand how the code is structured.

### Tools you may require

The following tools are only our recommendations; you are free to use anything you feel comfortable with.

Editor: Dreamweaver MX

Browser: Firefox

System: Windows Platform

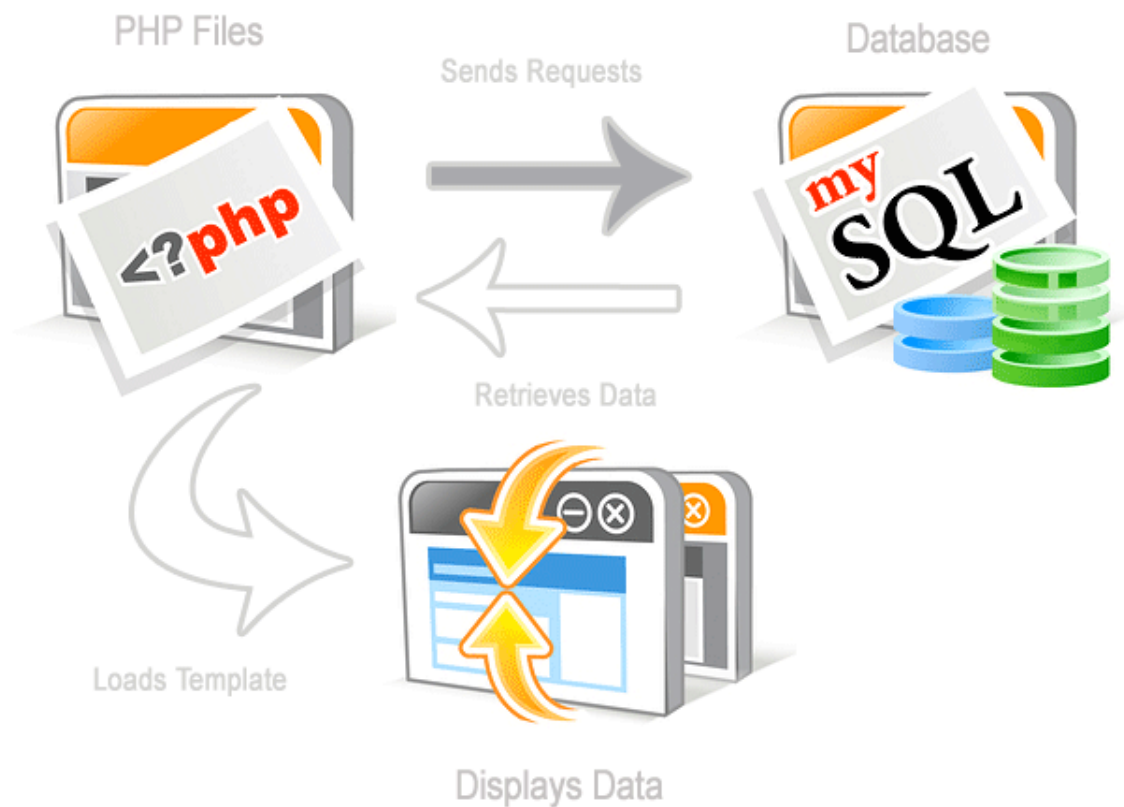
Graphic Editor: Photoshop 7+

## Getting Started

### Understanding the software and framework.

Before you can dive in and start tearing away at the software files, it's a good idea to learn how the system works, where and how the files are stored and called within system.

The diagram below shows the basic system overview.



## The eMeeting Framework

The eMeeting framework is based on an in house solution, developed by Mark Fail. The basic concept is to enable excellent performance and flexibility whilst maintaining good code and file structure.

The system works on three main layers.

### 1. PHP code structure (Input)

The PHP code is executed first before the HTML code is displayed, the PHP talks with the database to send and receive data based on the users operations.

### 2. MYSQL Database (store)

Stores all member data within relational database tables.








### 3. HTML Template (Output)

The template files are 80% html and css, there are a little snippets of PHP code which are used to loop data and display dynamic content.






## File Structure and Locations














The following information is designed to provide a reference sheet allowing you to quickly and easily locate files used throughout the system. Not all files are listed below as those files may not be “required” by the system or no longer used. The files below are the only files you would need to be concerned about when developing or editing the software.

### Root Directory




File Path	Description
 Root / Ppublic_html/	This is the root folder of your server or directory of install. This folder name is not included in the software, its merely to represet the folder of install.
 Inc/	Stores included data for the template.
 Install/	Stores the installation files
 Newadmin/	Store the admn area files.
 Plugins/	Store the software plugins
 Uploads/	Store the member files
 Images	Store the system and template graphics.

### Inc Folder

File Path	Description
 inc/	
 API/	Stores an API class file
 Classes/	Store system class files
 Css/	Store system css (stylesheets) files
 Exe/	Stores Software adons



 Func/	Stores system function files
 Js/	Stores system javascript files
 Langs/	Stores system language files
 Lib/	Stores Validation image
 Payment/	Stores payment gateway files
 Templates	Stores template files
 XML	Store XML feeds
 Ajax/	Store ajax files
 Config.php	Main system configuration file
 Config_db.php	Database connection file
 Error404.php	Error catching file
 Config_template.php	Stores all colours and home page content
 Tb.php	Used to display the website images

## Plugins Folder

File Path	Description
 plugins/	
 plugins/	Stores plugin files and folders
 config_plugins.php	Tells the system which plugin files to be loaded.

## Uploads Folder

File Path	Description
 uploads/	
 files/	Store Member template and admin files.
 Images/	Store member image files.
 Music/	Stores member music files.
 Thumbs/	Stores member image thumbnails.

 Videos/	Stores member video files.
 Tb.php	Generates the image thumbnails

## Images Directory

### File Path

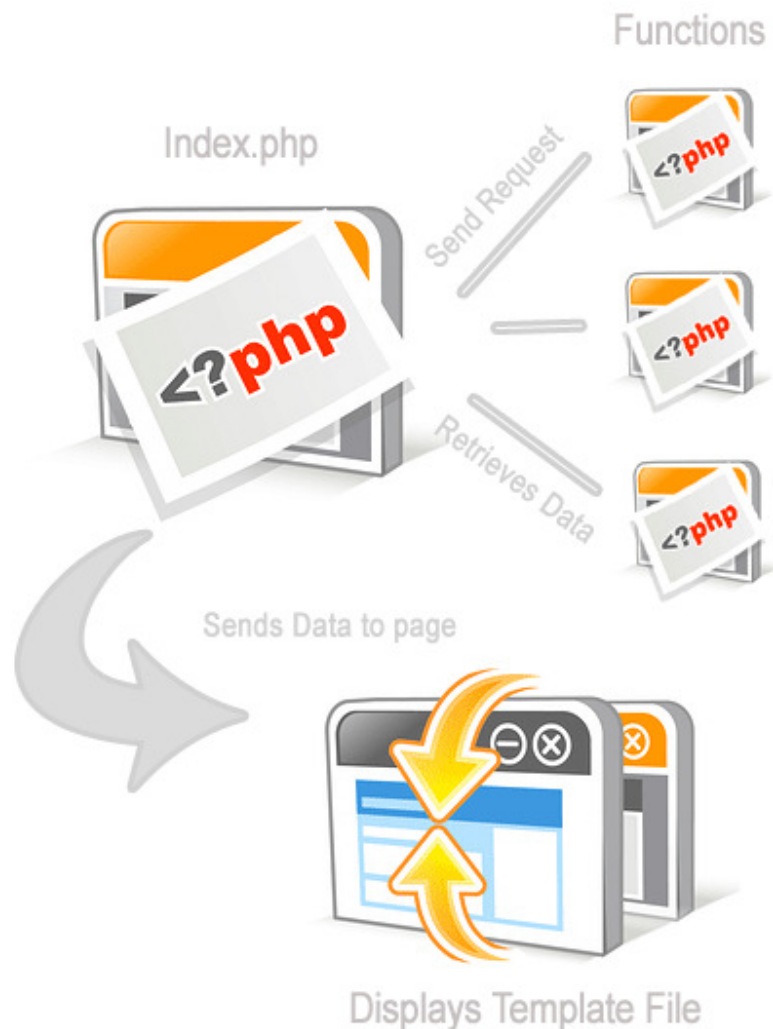
File Path	Description
 DEFAULT/	Stores all system graphics
 /template_folder_names/**	Stores all template graphics

\*\* Template Folder Names is not a folder, its labelled above simply to explain that there will be many different folder names such as “v8\_1”, “v8\_2” etc. These represent the folder names for each of the templates. Inside each of these folders are the files used by that template.

## How the system works.

The software works in the following way; it runs the index.php file which makes all the calls to the necessary php functions based on the page request. The php functions store lots of data in arrays which are then displayed on the HTML pages. Its that simple!

Here's a quick diagram;



## How are the pages determined?

There are two main page requests sent to the system. First you have a “main page” which we have given the name “dll” and there is a sub page which displays sections of the main page. For this we have used the name “sub”.

A sample URL would be: <index.php?dll=messages&sub=inbox>

Looking at the above example we can see the main page we are calling is the messages page and the sub content within this page is the “inbox”. What this allows us to do is group lots of different pages together based on the main page content. So for example we have a number of pages that relate to messages such as “outbox”, “trash”, “sent” and “compose” by changing the url slightly you could call any one of these sub pages.

If you are adding new pages you simply link to them in the same way. For example, if you add a new “main page” called “test” with an inner subsection called “testing” we would call this page as below;

<index.php?dll=test&sub=testing>

## How do I add a new page?

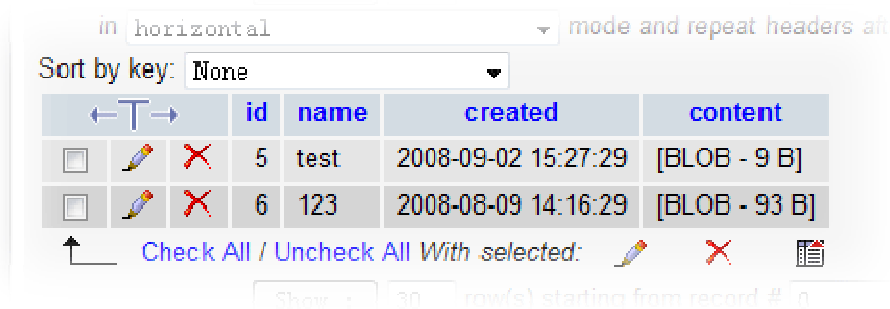
Adding new pages is easy but there are two ways of doing it so here goes!

### 1. Adding a page via the admin area

By adding a page via the admin area the page data is stored in the database for easy editing and retrieval, usually these types of pages are information based only, for example a links page, promotional offers page etc. Its not recommended to add pages you want to develop on simply because there is not "file", its just data in the database and its often nice to refer back to a file incase you want to edit it offline.

The pages added to the database are stored in the table: `template_pages`

The table structure looks like this:



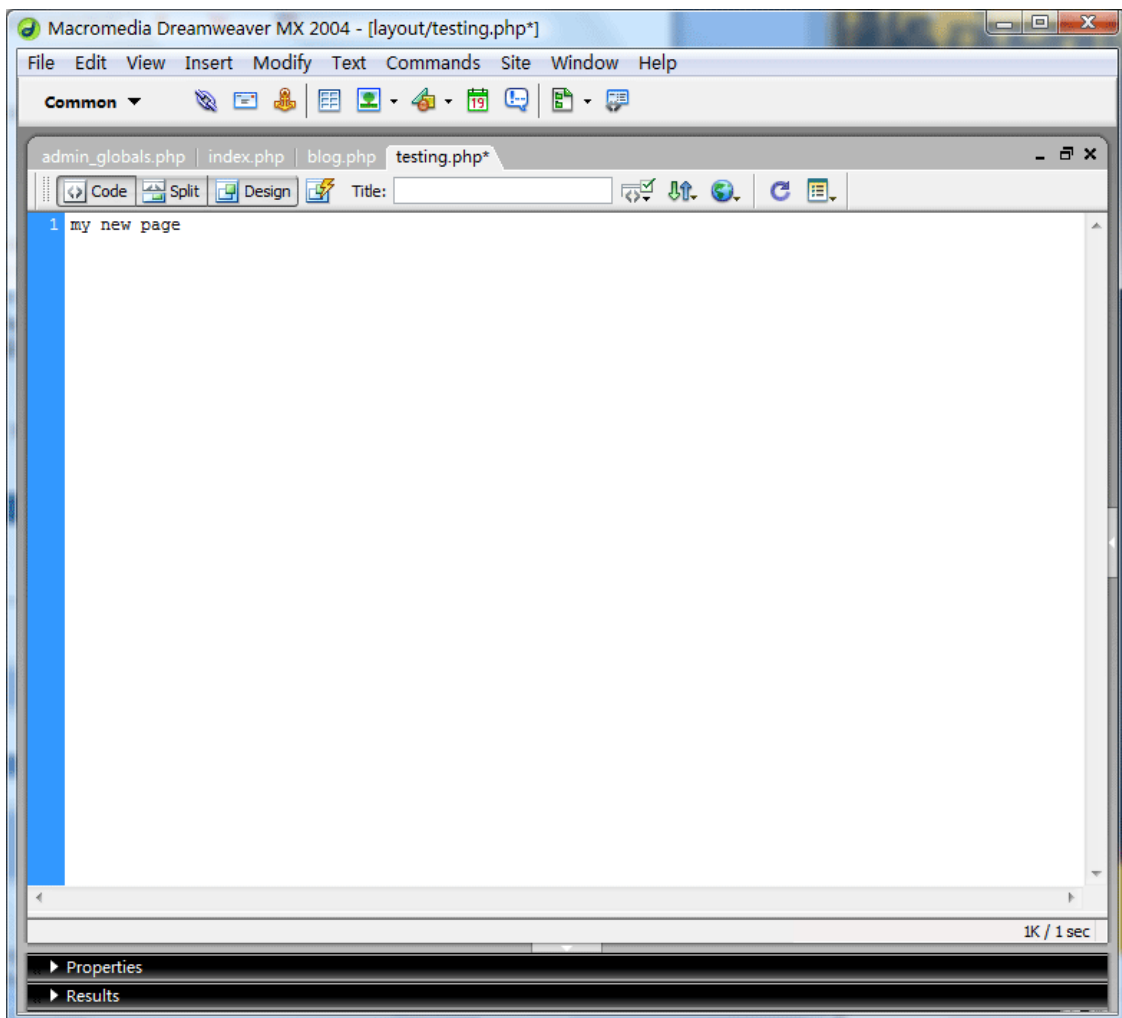
The screenshot shows a database table view for 'template\_pages'. The table has columns for 'id', 'name', 'created', and 'content'. There are two rows of data. The first row has id 5, name 'test', created '2008-09-02 15:27:29', and content '[BLOB - 9 B]'. The second row has id 6, name '123', created '2008-08-09 14:16:29', and content '[BLOB - 93 B]'. The interface includes a 'Sort by key' dropdown set to 'None', a 'Check All / Uncheck All' button, and a 'Show 30 row(s) starting from record # 0' indicator.

	id	name	created	content
<input type="checkbox"/>	5	test	2008-09-02 15:27:29	[BLOB - 9 B]
<input type="checkbox"/>	6	123	2008-08-09 14:16:29	[BLOB - 93 B]


Its nothing special, the page has a nice, time is was created and the contents of the page.

## 2. Adding pages manually

Ok now here goes, adding pages manually requires two things. Firstly you need to create a blank PHP document with only the words “my new page” at the top. (add other content if you like but for this example it makes it easier) there is no need for html headers, title tags, meta tags etc as these will be inherited by the template. There must be no spaces in the file name, so for this example we will be creating a document called “testing.php” and it would look something like this.



Now you upload this file via FTP or your file manager tool to the following directory.

File Path	Description
 Inc/templates/layout/	All inner page content is stored here. Its where all the template pages can be found.

Ok, now the page is loaded into the system, we need to do one other thing to ensure it inherits all of the page design from the headers and footers.

We need to edit the index.php file!

I know this may sound alittle horrifying but lets explain how it works so we can understand why.

Firstly as we are developing the system and not just adding pages, this allows us to setup new pages within the system file structure, as explained previously, pages added into the database are merely therefore for displaying a single page of content without any real functionality. By adding the page to the index.php file we can better understand the code logic and begin building better systems.

Seondly it allows us to keep all the code tidy, one file loads the template page and one file displays it. Its that simple.

Ok, now all that being said, lets learn how to add the new page code to the index.php file.

## Hardcoding your pages

You need to now open up your index.php file and learn how and where to add new page content. This is what were looking for.

```
20     $menu_nav_name = $sub_page;
21     ////////////////////////////////////////////////////
22     // LOAD TEMPLATE CONTENT AREA
23     ////////////////////////////////////////////////////
24     switch($page){
25
26         /**
27         * Page: Home / Index Page
28         *
29         * @version 8.0
30         * @created Fri Jan 18 10:48:31 EEST
31         * @updated Fri Jan 18 10:48:31 EEST
32         */
33         case "index": {
34
35             $Index_Page_flag=1;
36             $HEADER_SINGLE_COLUMN = 'yes';
37             require_once('inc/func/func_index_');
38
39         } break;
40         /**
```

To be more precise, we are looking for the switch stament, shown above as:

```
Swtch($page){
```

The system works by looking through all the switch cases until it fines one that can handle the page request. So in the above example you can see the code;

```
case "index": {

    $Index_Page_flag=1;
```

```
$HEADER_SINGLE_COLUMN = 'yes';  
require_once('inc/func/func_index_page.php');  
  
} break;
```

This is the hardcoded version of the home page of our website. What we are going to do is add a new switch call right under the main switch statement. It should look something like this:


```
220     $menu_nav_name = $sub_page;  
221 ///////////////////////////////////////////////////////////////////  
222 // LOAD TEMPLATE CONTENT AREA  
223 ///////////////////////////////////////////////////////////////////  
224 switch($page) {  
225  
226     case "testing": {  
227  
228  
229     } break;  
230     /**  
231     * Page: Home / Index Page  
232     *  
233     * @version 8.0  
234     * @created Fri Jan 18 10:48:31 EEST 2008  
235     * @updated Fri Jan 18 10:48:31 EEST 2008  
236     */  
237     case "index": {  
238  
239         $Index_Page_flag=1;  
240         $HEADER_SINGLE_COLUMN = 'yes';  
241         require_once('inc/func/func_index_page.php');  
242  
243     } break;
```

The example above shows we have added the following code right AFTER the switch call.



## Adding and editing page content

Ok, so now we have managed to successfully create a new page ( I hope! ) that is hardcoded within the system allowing us to use this later. Or for those who have jumped sections and want to know where to edit the pages, all pages are stored in the following directory;


File Path	Description
 Inc/templates/layout/	All inner page content is stored here. Its where all the template pages can be found.

If you open up this folder you will find a list page file names which directly relate to the page name. So for example our newly created page that we added in the section above, is called “testing.php” which is loaded by the page call “testing”. (index.php?dll=testing).

Or if you’re trying to edit the messages page, the page name would be called “messages.php” which is loaded by the page call “messages”. (index.php?dll=messages)

### Exceptions to this rule ( editing the header and footer )

There is one exception to this rule which is when your loading the template header and footer pages. These are stored within the folder name of the template you’re using, so if we are using the template called “v8\_1” these files will be found in the following directory;

File Path	Description
 Inc/templates/v8_1/	Location of the header and footer files for the template v8_1

## Adding Content

You can basically add any content you want to your pages, there is no restrictions just keep in mind that if you use an editor such as front page or dreamweaver that the folder path that any added files are store will default to the root folder as it's the index.php file that will load this page.

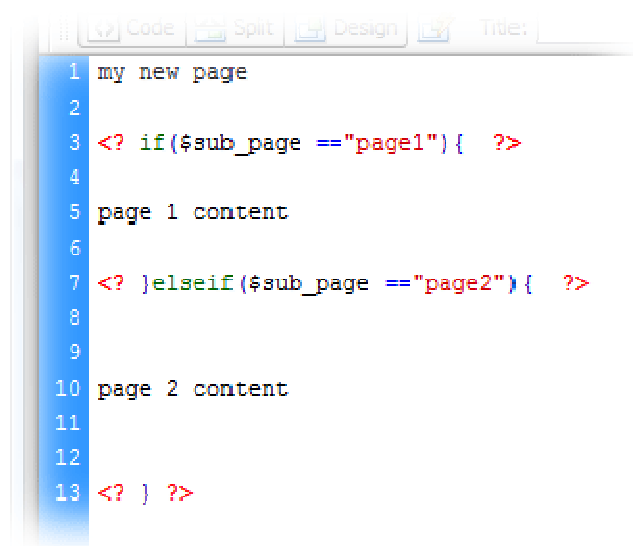
## Creating Sub pages

In this section we are going to learn how to divide our main page into sub pages that can be called using the URL string. "sub". So for example:

```
index.php?dll=testing&sub=page1
```

the above URL string will load the testing.php file and look for a sub page called "page1". This allows us to create one file and split it into different sections.

A sample file may look like this.

A screenshot of a code editor window. The window has a title bar with "Code", "Split", "Design", and "Title:" buttons. The code is as follows:

```
1 my new page
2
3 <? if($sub_page == "page1"){ ?>
4
5 page 1 content
6
7 <? }elseif($sub_page == "page2"){ ?>
8
9
10 page 2 content
11
12
13 <? } ?>
```

The example above is the testing.php file create earlier split into two pages, we now have page 1 and page 2.

To ensure security, \$sub\_page is a variable cleaned from the URL request and holds the same value as &sub=page1 (\$\_GET['sub']).

That's basically everything you need to split your page into sub pages.

## Posting Form Data

Just like within the URL data, you can return to pages and call new ones when submitting forms. All you have to do is include the pages values within the form element.

Code Example Below

```
<form method="post" action="index.php">  
  
<input name="do" type="hidden" value="addpost" class='hidden'>  
<input name="do_page" type="hidden" value="testing" class="hidden">  
<input name="sub" type="hidden" value="blog" class="hidden">  
  
.... CONTENT HERE  
  
</form>
```

What do you think would happen if you submitted this form?

Ok, well basically the form has three data elements which are most important.

```
<input name="do" type="hidden" value="addpost" class='hidden'>
```

This value tells the system what function call to do, so in the example above we are going to “addpost” or literally “add a new post”. You can create your own

PHP functions within your newly create pages to handle these requests. See the section below on handling submitted data.

```
<input name="do_page" type="hidden" value="testing" class="hidden">
```

This value tells us what page to send the request to. So in this case we are sending the request to the “testing” page and functions that are used by this section of the website.

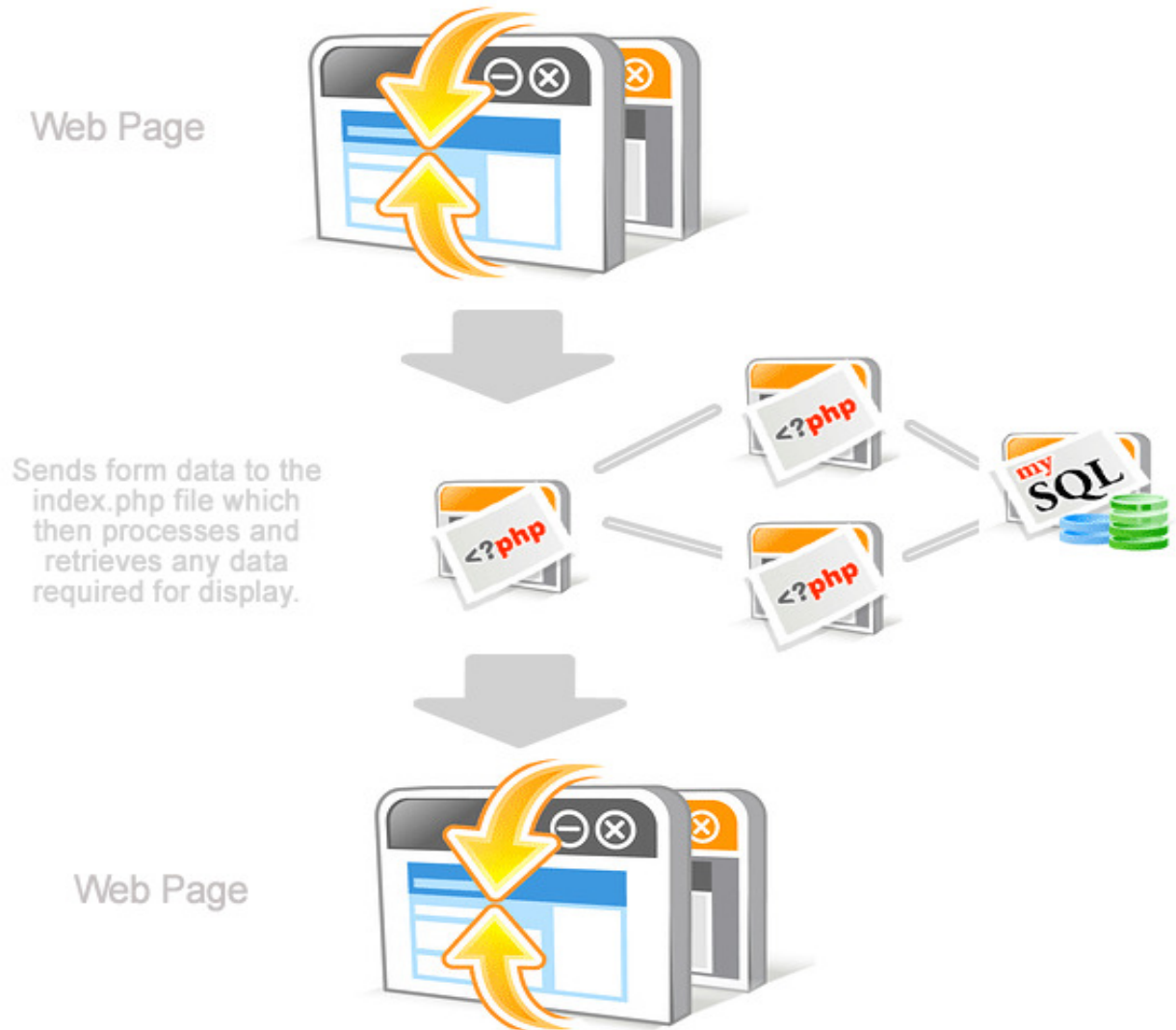
```
<input name="sub" type="hidden" value="blog" class="hidden">
```

Finally we have a sub element. This basically tells the system which page to return to once the data has been processed. So in this case we will be returning back to the “blog” page after the form has been submitted and processed.

## **How is data send and received by the system?**

Ok, as the system has only one main file its easy to figure out where the data is being sent. The index.php file within the root of the software takes all of the data and handles it based on your request. So, if I submit a form on the contact page or upload a photo, the page that handles the requests always going to be the index.php file. Why? Well it makes everything a whole lot easier and more streamline! That’s why!

Diagram below on how form data is submitted.



## Handling Submitted Data

Data submitted by forms is handled in the same way as everything else. Its passed to the index.php file, the index.php file locates the name of the page being loaded and then looks for some functions to handle the request.

```
222 // LOAD TEMPLATE CONTENT AREA
223 ////////////////////////////////////////////////////
224 switch($page) {
225
226     case "testing": {
227
228         if(isset($_POST['do'])){
229             handle function here..
230         }
231
232     } break;
233     /**
234     * Page: Home / Index Page
```

The above diagram shows the switch call we added previously, but with a few added lines of code;

```
if(isset($_POST['do'])){
    handle function here..
}
```

All this simply does it checks when the page is loaded for any form data, data send to the index.php file when a form is submitted. We can then handle the form data any way we like.



## **MYSQL Structure**

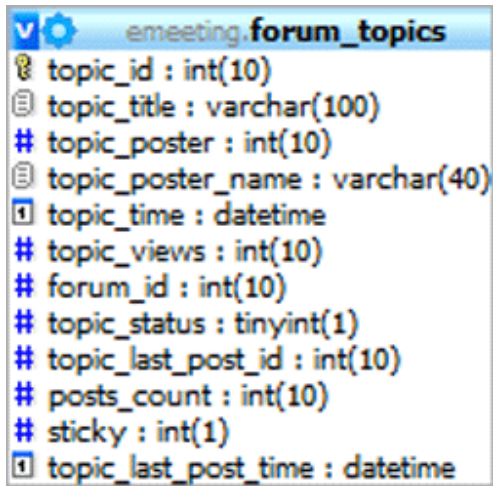
eMeeting software uses a MYSQL database to store all of the member information. The database DOES NOT store information about the website setup, servers settings etc. All system data is stored within the config files.

## Making Database Queries

Making Database queries is actually very simply, all the functions have been loaded by the software so you can just need to think about the query your going to create.

## Data Dictionary

The following is a list of tables used by the system and their attributes.



```
emeeting.forum_topics
topic_id : int(10)
topic_title : varchar(100)
topic_poster : int(10)
topic_poster_name : varchar(40)
topic_time : datetime
topic_views : int(10)
forum_id : int(10)
topic_status : tinyint(1)
topic_last_post_id : int(10)
posts_count : int(10)
sticky : int(1)
topic_last_post_time : datetime
```

## Retrieving a single row of data

Lets say we want to create a new query that gets the email address for the member who's member name is "Alan".

The query would look something like this;

```
SELECT email FROM members WHERE username ="Alan" LIMIT 1
```

This is basic SQL syntax which finds the email value from the table "members" with the username "Alan".

The PHP syntax would look something like this;

```
<?
$Data = $DB->Row("SELECT email FROM members WHERE username ='Alan'
LIMIT 1");
print $Data['username'];
?>
```

```
<?
3 $Data = $DB->Row("SELECT email FROM members WHERE username ='Alan' LIMIT 1");
10 print $Data['username'];
11
?>
```

This query will do a single row select based on the SQL above.

## Retrieving More than one value

If you want to retrieve all the members email addresses, you can do the following query;

```
SELECT email FROM members
```

The PHP syntax would look something like this;



```
6
7 <?
8
9 $Data = $DB->Query("SELECT email FROM members");
10 while( $value = $DB->NextRow($Data) )
11 {
12     print $value['email'];
13 }
14
15 ?>
16
```

```
<?
$Data = $DB->Query("SELECT email FROM members");
while( $value = $DB->NextRow($Data) )
{
    print $value['email'];
}
?>
```

Notice the slight difference in the PHP syntax, this time we're getting lots of data from the database so we need to loop each row of data found and display the values.



## The Config file

There are two major config files used within the eMeeting software. These are;

File Path	Description
 Inc/config.php	This config file stores all the website constants which determine the display of your website.
 Inc/config_db.php	This config file stores your database configuration file, if you have any problems with database connectivity or you move servers you will need to consult this file to ensure all username and passwords are correct.

## Language Files

All of the software language files are stored in the following directories;

File Path	Description
 Inc/langs/	Member area language files
 Newadmin/inc/langs/	Admin area language files

To create your own language file simply copy one of the existing language files, rename it and then translate the content within the file.


### Translation example;

```
"_join" => "Not yet a member?",
"_join1" => "Its free to join so what you waiting for!",
"_join2" => "Signup Now!",
```

You would only translate the English sentences, so a translation to german might be;

```
"_join" => "Noch kein Mitglied?",
"_join1" => "kostenlose Registrierung!",
"_join2" => "Jetzt registrieren!",
```

The language file flags are stored;

File Path	Description
 Images/languages/	Stored the flags.