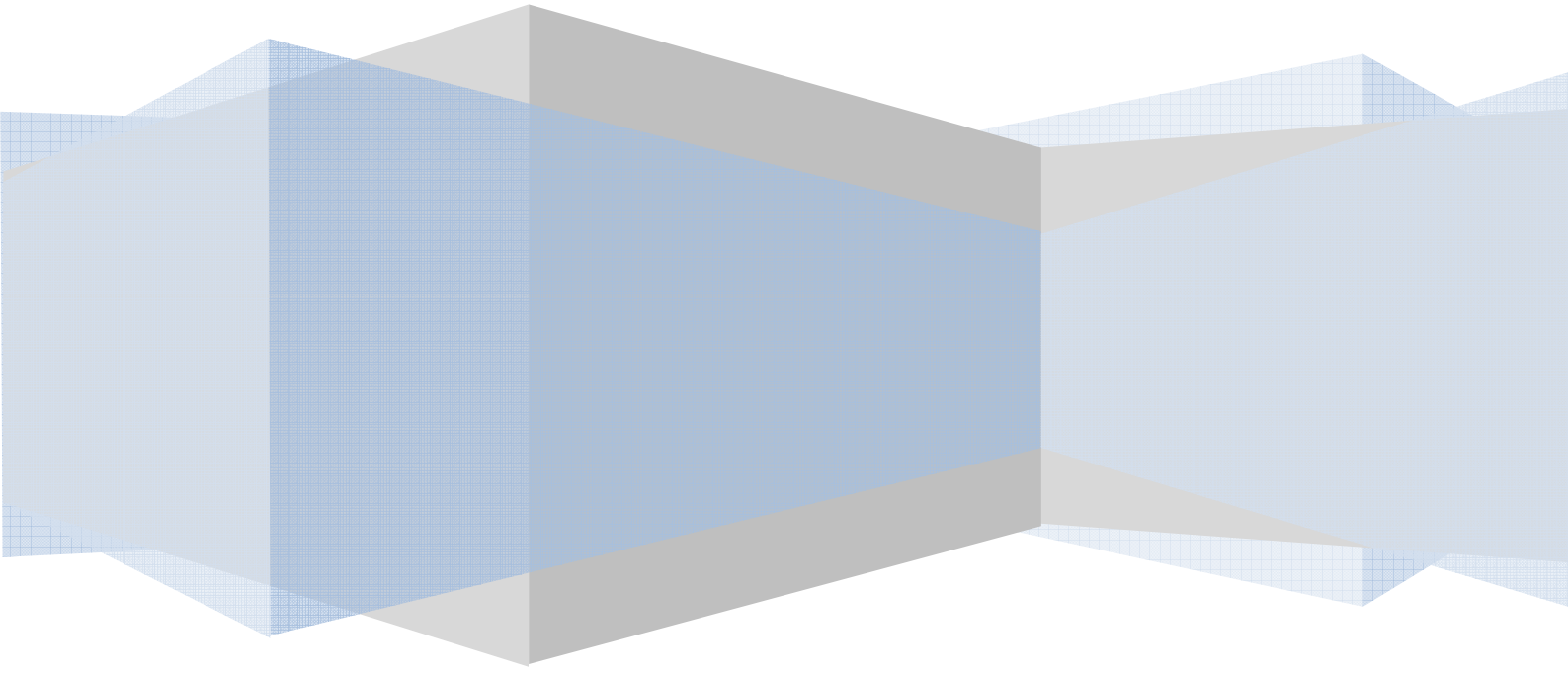
Maintenance Document

Teaching Management System



Contents

1. Application Framework

The Teaching Management System is built on a framework that consists of various modules. When the system is accessed via a browser, the system access index.php, the main controller of the system, which will load the page the user requested, automatically connect to the database and authenticate the user.

1.1 Folder Layout

Under the main directory there are several sub-directories:

- `images` – any images used in the application are stored within the images folder.
- `includes` – files used in the running of the framework, including setting up the user, loading modules, and the default HTML design.
- `javascript` – any modules that require JavaScript store their code in this folder. It also includes the JQuery library plus any required Jquery addons.
- `modules` – where the system's modules run from. Each module has its own directory, with a file of the same name in the directory, which is a class with the same name.
- `utilities` – any system utilities can be stored in this directory./

The main folder also includes several important files:

- `.htaccess` – Apache HTAccess rules that establish the clean URLs
- `ajax.php` – Primarily used for auto-complete fetching of staff names
- `config.php` – Where configurable settings reside (title, semester, etc)
- `index.php` – Main controller of the application
- `initiate.php` – System settings, MySQL connection, etc
- `print.css` – CSS stylesheet for printing
- `style.css` – Main CSS stylesheet.

1.2 Module Layout

A module in the Teaching Management System must be laid out as follows. A module will have it's own directory, which will be the module name. Inside the directory, you must create a file with the same name of the directory, appended with `.php`, and it should be a class with the same name, and extend the `FW_Module` class.

For instance, if you wished to create an example module, you would create a directory called `example`, and then a file named `example.php`, which was set up as follows:

```
<?php
class example extends FW_Module
{
}
?>
```

Modules are made up of several classes. There is the default class, which is the module with the same name as the directory, and then you can create any number of classes within this module.

Classes must have the same name as their filename, or they will not run. Inside each class, you can have various functions you want to run. If no function is specified in the URL (see later), the controller will attempt to run a function called `__default` (two underscores).

By default, any functions you create in a class, other than `__default` are considered private and not accessible from the web. To make a function accessible, you must add it to an array in the class file called `$public`. Please consider this example class:

```
<?php
class example extends FW_Module
{

	var $public = array("isPublic", "accessNotPublic");

	function __default()
	{
		echo "default function";
	}

	function notPublic()
	{
		echo "cannot access this directly";
	}

	function isPublic()
	{
		echo "can be accessed directly. ";
	}

	function accessNotPublic()
	{
		$this->notPublic();
	}

}

?>
```

In this example, no function would print “default function”, notPublic() would present an error, isPublic would print “can be accessed directly” and accessNotPublic would run notPublic to print out “cannot access directly.”

There are two helper functions available in the FW_Module class:

`__authenticate()`

Authenticate is called by the system to check the user is allowed to access this class. By default, the authenticate function will check if a user’s access level is at least 1. If you want all users to access this, even if they are not logged in, you can override this function just to return `true`. This is useful for the logon page.

If you wanted the function only to allow users with access level 2 to access this class (administrator users), you could override the function as follows:

```
function __authenticate() {
      // Check only administrators can use this page
      return ($this->user->getLevel() == 2);
}
```

`__buildlink($module, $class, $function)`

This function will automatically build the URL required to access a page with the module, class and function you specify. The only required variable is module, if you miss class and function, the defaults will be supplied. The default for class is the class with the same name, the default for function is `__default`.

1.3 URLS

The URLs used to access the different pages in the application are structured as follows:

- http://website/ - access main page
- http://website/example - access the module called `example`
- http://website/example/test - access the module called `example` and the class called `test`
- http://website/example/test/function - access the module called `example` and the class called `test` and the function called `function`

The `$this->__buildlink` function described above should be used in the application to construct these URLs.

1.4 Users

There are three types of users in the system, with different access levels.

- 2: Administrator
- 1: Staff
- 0: Guest

Only administrators and staff users can log on, unless you override the `__authenticate` function. All users are stored in the same table. When using the system, in a module, you will have access to the currently logged on user in the `$this->user` class. This is of type `FW_User` and provides the following functions.

`loggedIn()`

This function will return a Boolean on whether the user is logged in or not. For instance, if a user is not logged in, there will still be a user class, only it will not be substantiated.

`getUsername()`

This will return the user's username.

`getName()`

This will return the user's full name.

`getUserID()`

This will return the user's unique ID number in the database. Useful when creating links to other database entities.

`getLevel()`

This will return the user's access level. This is useful when working on the `__authenticate` function to check if the user is allowed to access this page.

1.5 HTML

A module will also have access to the standard HTML design through the `$this->html` object. This works in a header/footer fashion, when you want to start the page HTML, you can call `$this->html->header();` and to end the HTML, you can call `$this->html->footer();`. The following functions are available in the HTML object::

`header($title)`

This will print out the HTML header, including the starting tags, the <head> tags, the top title bar and the menu, plus print a logout form. You can optionally specify a title, but otherwise, the system will use a blank title.

`footer()`

This will print out the end of the HTML.

`setTitle($title)`

Set the title of the page. This **must** be called before using `header`.

`setExtra($extra)`

Set any extra HTML content you wish to appear in the <head> area. This can be useful for including extra JavaScript or CSS. This **must** be called before using `header`.

`setExtraMenu($extramenu)`

Set any extra HTML you wish to appear in the menu. This is useful when you want extra links to appear in the menu for just this page.

If you wish to change the HTML design, you must edit the `includes/HTMLManager.php` file, and modify the various functions you wish to change, most likely in the header and footer functions.

2. Default Modules

This section documents the modules available in the Teaching Management System:

- Change Password (`changepassword`)
  This is a simple tool to allow any logged in user change their password.
- Errors (`errors`)
  This is a system-used module that prints out errors when the system encounters problems.
- Home (`home`)
  The default module when a user has logged in. This includes all the links to the various sections of the system, and is the base for the navigation.
- Login (`login`)
  Module that provides a login form, and logout options.
- Module Runs (`moduleruns`)
  Where the main module run data is entered, as well reports, archive and module timetables.
- Settings (`settings`)
  Where the system settings can be changed. This includes the title, year, dates etc that the system will depend upon.
- Setup Courses and Modules (`setup`)
  Where the administrator users can create modules, courses, module runs. These are split up between the class files.
- Teaching Load (`teachingload`)
  Various reports for teachers, as well as reports for administrators that allow them to see reports for each individual staff user.
- Users (`users`)
  Where administrator users can create, edit or delete any user accounts.
- Weightings (`weightings`)
  Where administrators can change weightings, or add/delete weightings.

3. Database (MySQL)

The database is composed of thirteen tables:

- `approved` – Linker table between `module_runs` and `courses`.
- `courses` – Stores the individual course names
- `directors` – Linker table between `users` and `courses`
- `module_runs` – Where details of a particular module running in a semester are stored. A module run will link to the `module` table. Also links to the `users` table to store the module coordinator.
- `sessions` – Where individual lectures/tutorials are stored for a particular module run.
- `sessions_staff` – Linker table between `sessions` and `users`.
- `supervision` – where a staff members supervision details are stored. Links to the `user` table.
- `users` – Where each user (admin, staff or guest) is stored.
- `worktypes` – Where weighting (work types) are stored.

3.1 SQL Schema

```
DROP TABLE IF EXISTS `approved`;
CREATE TABLE `approved` (
  `courseid` int(10) unsigned NOT NULL,
  `modulerunid` int(11) NOT NULL default '0',
  `optional` tinyint(1) NOT NULL,
  PRIMARY KEY  USING BTREE (`courseid`,`modulerunid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `courses`;
CREATE TABLE `courses` (
  `id` int(11) unsigned NOT NULL auto_increment,
  `name` tinytext NOT NULL,
  PRIMARY KEY  USING BTREE (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `directors`;
CREATE TABLE `directors` (
  `courseid` int(10) unsigned NOT NULL,
  `userid` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`courseid`,`userid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `involved`;
CREATE TABLE `involved` (
  `id` int(11) NOT NULL auto_increment,
  `module` int(11) default NULL,
  `staffname` varchar(32) default NULL,
  `role` varchar(6) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
DROP TABLE IF EXISTS `module_runs`;
CREATE TABLE `module_runs` (
  `id` int(11) NOT NULL auto_increment,
  `modulecode` varchar(10) NOT NULL,
  `semester` varchar(6) NOT NULL default 'Autumn',
  `year` int(4) NOT NULL,
  `coordinator` int(11) default NULL,
  `assessed` int(2) default NULL,
  `nonassessed` int(2) default NULL,
  `changes` text,
  `changes_staffing` text,
  `runnextyear` tinyint(1) NOT NULL default '0',
  `new` tinyint(1) unsigned NOT NULL default '1',
  `lastupdated` datetime default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 ;

DROP TABLE IF EXISTS `modules`;
CREATE TABLE `modules` (
  `code` varchar(10) NOT NULL,
  `name` varchar(60) NOT NULL,
  `disabled` tinyint(1) unsigned default '0',
  PRIMARY KEY  (`code`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `sessions`;
CREATE TABLE `sessions` (
  `id` int(11) NOT NULL auto_increment,
  `modulerunid` int(11) default NULL,
  `week` int(2) NOT NULL,
  `title` tinytext NOT NULL,
  `day` tinyint(1) unsigned default NULL,
  `start-time` time default NULL,
  `end-time` time NOT NULL,
  `length` double(4,2) default NULL,
  `type` varchar(32) default NULL,
  `room` varchar(20) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `sessions_staff`;
CREATE TABLE `sessions_staff` (
  `sessionid` int(11) unsigned NOT NULL,
  `userid` int(11) unsigned NOT NULL,
  `modulerunid` int(10) unsigned default NULL,
  PRIMARY KEY  (`sessionid`,`userid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `supervision`;
CREATE TABLE `supervision` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `userid` int(10) unsigned NOT NULL,
  `lead` tinyint(1) NOT NULL,
  `hours` tinyint(3) unsigned NOT NULL default '12',
  `student` tinytext NOT NULL,
  `year` int(10) unsigned NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `id` int(11) NOT NULL auto_increment,
  `username` varchar(32) NOT NULL,
  `password` varchar(32) NOT NULL,
  `name` varchar(32) NOT NULL,
  `lastname` varchar(32) NOT NULL,
  `accesslevel` tinyint(4) NOT NULL,
  `resetkey` varchar(32) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `worktypes`;
CREATE TABLE `worktypes` (
  `id` int(3) NOT NULL auto_increment,
  `type` varchar(32) NOT NULL,
  `weighting` double(4,2) NOT NULL,
  `description` tinytext,
  `scheduled` tinyint(1) unsigned NOT NULL default '1',
  `disabled` tinyint(1) default '0',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```