Affero -- Developer Documentation

Contents

Package default Procedural Elements

manage_index.php

- **Package** default

include **'manage/time.php'** *[line 38]*

include **'manage/skill.php'** *[line 33]*

include **'manage/area.php'** *[line 28]*

dbsetup.php

- **Package** default

include **'app/libraries/database.lib.php'** *[line 8]*

include **'app/libraries/utility.lib.php'** *[line 7]*

include **'app/config.php'** *[line 6]*

Package affero Procedural Elements

config.php

Affero Configuration File

This file is used to set all the configurable settings of defero.

- **Package** affero
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 Willaim Duyck
- **License** MPL

affero.php

Affero Controller

This file contains all the controls for the front end of affero, from the recomendation engine to the form that gets user details/interests/skills

- **Package** affero
- **Sub-Package** controllers
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

dashboard.php

Dashboard Controller

This file contains a collection of tools for the running of the affero dashboard

- **Package** affero
- **Sub-Package** controllers
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

manage.php

Manage Controller

This file contains a collection of tools for managing the areas of contribution, skills, and time requirements that are used when generating recomendations on the front end of affero

- **Package** affero
- **Sub-Package** controllers
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

user.php

User Controller

This file contains a collection of tools for logging in/logging out users as well as creating them, modifying them, and deleting them from the database.

- **Package** affero
- **Sub-Package** controllers
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

Package affero Classes

Class Affero

[line 23]

Affero

This class provides all the processing behind the front end which is both public and user access able.

- **Package** affero
- **Sub-Package** controllers

void function Affero::index() *[line 31]*

index

Loads the needed information for the main frontend user interface.

void function Affero::out($args) *[line 43]*

Function Parameters:

- **$args**

out

Grabs all the information needed for the metrics and then fwds the user off to the correct website

Class Dashboard

[line 23]

Dashboard

This class provides all the processing behind the dashboard which is both public and user access able.

- **Package** affero
- **Sub-Package** controllers

void function Dashboard::index() *[line 31]*

index

This loads the view as well as the data that drives it

Class Manage

[line 30]

Manage

This class provides all the processing behind the views that allow users to make changes to areas of contribution, etc...

- **Package** affero
- **Sub-Package** controllers
- **TODO** Rethink edit functions
  - remove area_edit
  - remove area_edit view
  - refactor area to control add/edit/delete all in one
  - refactor area_add to work for both add and edit

void function Manage::area() *[line 274]*

area

This function handles editing/deleting/creating areas

void function Manage::index() *[line 38]*

index

This is the default page for all management tasks, it will likely contain some warnings and links to different common tasks

void function Manage::skill() *[line 93]*

skill

This function handles editing/deleting/creating skills

void function Manage::timeRequirement() *[line 192]*

timeRequirement

This function handles the editing/deleting/creating of time requirements

Class User

[line 27]

User

This class deals with all user management functions from logging users in to creating/removing their accounts.

- **Package** affero
- **Sub-Package** controllers
- **TODO** user class items
  - create add user function

void function User::create() *[line 56]*

create

this function creates a user, it does so by checking the provided token, unencrypting it if it is valid, and then asking the user to provide a username and password. (the decrypted token is the users email)

- **Access** public

void function User::delete() *[line 356]*

delete

This function is responsible for the deleting users. It will ask the user for confirmation that they wish to remove their account as well as check that they have the correct password so as to avoid someone removing their account by accident, and for added security.

On success this function will run the logout process, and redirect the user to the publicly available dashboard with a notice informing them of success.

Only logged in users may use this method

- **Access** public

void function User::index() *[line 38]*

index

provides a page to check if a user is logged in and routes them to either the backend dashboard OR the login page depending on the result

- **Access** public

void function User::invite() *[line 169]*

invite

provides a way for existing users to invite new users

- **TODO** invite system
  - refactor the code
- **Access** public

void function User::login() *[line 419]*

login

this function is the outer face of the login system. It handles both get and post requests. When receiving a get request it loads the login form and on a post request uses the input class to get the username and password submitted and hand it off for processing to process_login

- **Access** public

bool function User::logout() *[line 484]*

logout

this function destroies user session information and logs them out of the app.

on success will redirect to the backend dashboard

bool function User::process_login($username, $password) *[line 455]*

Function Parameters:

- *string* **$username** the username of the user to login
- *string* **$password** the password of the user to login

process_login

this function is responsible for the creation of sessions relating to users PROVIDED that

the function receives valid credentials.

- **Access** private

void function User::settings() *[line 259]*

settings

provides the ability to change user settings such as password and email address

- **Access** public

index.php

Affero

This file contains all the url routing and class loading for the entire application. It is essentially the core of the application.

- **Package** affero
- **Sub-Package** core
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** MPL 1.1/LGPL 2.1/GPL 2.0 William Duyck

include **dirname(__FILE__).'/asset/error/405.html'** *[line 163]*

include **dirname(__FILE__).'/asset/error/404.html'** *[line 171]*

void function __autoload($class) *[line 22]*

Function Parameters:

- **$class**

__autoload

this function will attempt to automatically load files/classes as and when they are needed rather than consume lots of memory and loading all classes

Class Controller

[line 56]

Controller

This class controls all tasks relating to modifying areas of affero. It does some minor url routing tasks, and provides all the key libraries to the controllers to routes to.

- **Package** affero
- **Sub-Package** core

Controller::$config

mixed = [line 59]

- **Access** protected

Controller::$database

mixed = [line 61]

- **Access** protected

Controller::$input

mixed = [line 65]

- **Access** protected

Controller::$utility

mixed = [line 63]

- **Access** protected

Controller::$view

mixed = [line 67]

- **Access** protected

Constructor *void* function Controller::__construct() *[line 75]*

__construct

this function sets up the backend requirements such as the configuration, database connection, etc...

void function Controller::GET($args) *[line 102]*

Function Parameters:

- **$args**

GET

This function routes all HTTP GET calls to the relevant controllers based on the url provided

void function Controller::POST($args) *[line 145]*

Function Parameters:

- **$args**

POST

very similar to GET. So similar in fact that it just sends all its calls striaght to GET for convinience. This will likely be modified in the future.

database.lib.php

Database Abstraction Layer

This file contains all the code needed for interacting with the database directly and provides tools to make life simpler and make the rest of the coding quicker.

- **Package** affero
- **Sub-Package** libraries
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

glue.lib.php

GluePHP

Glue is a PHP micro-framework. It provides one simple service: to maps URLs to Classes. Everything else is up to you. The database, ORM, template engine and all other components are under your control. Glue just glues everything together.

In MVC terms, Glue is the URL Routing and Controller portion while you have total control over your choice of a Model and View layer.

- **Package** affero
- **Sub-Package** libraries
- **Author** Joe Topjian
- **Version** 1.0
- **Copyright** 2009 Joe Topjian
- **License** BSD

input.lib.php

Input Class

This file contains a collection of tools for sanitizing, and filtering user supplied data via $_GLOBAL's. It does so by unregistering globals, checking for xss type inputs, and standardising new lines.

- **Package** affero
- **Sub-Package** libraries
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

rest.lib.php

Rest

This file contains a collection of helper functions for the api

- **Package** affero
- **Sub-Package** libraries
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

utility.lib.php

Utility

This file contains a collection of utility methods that are not related to a particular area of the application.

- **Package** affero
- **Sub-Package** libraries
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** MPL 1.1/LGPL 2.1/GPL 2.0 William Duyck

view.lib.php

View

This file contains a collection of helper functions to make view creation easier and to remove redundancy. It also allows for easier theming of the system.

This library is slightly different to the others as it relies on the utility library

- **Package** affero
- **Sub-Package** libraries
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

Class Database

[line 28]

Database

Provides an easy way to run simple database queries with good optimisation and a reduction on database overhead caused by deletion of rows.

- **Package** affero
- **Sub-Package** libraries
- **TODO** DAL Rewrite
  - Rewite all methods to use the newer mysqli functions over the mysql ones

Database::$_handler

mixed = [line 31]

- **Access** private

Database::$_result

mixed = [line 33]

- **Access** private

Constructor *mixed* function Database::__construct([$host = null], [$database = null], [$username = null], [$password = null]) *[line 49]*

Function Parameters:

- *string* **$host** host server for the database
- *string* **$database** the database name to connect to
- *string* **$username** username to connect to the database with
- *string* **$password** the password for the database

__construct

Provides an alias for initilizing the class and connecting to a database straight away.

bool function Database::connect($host, $database, $username, $password) *[line 69]*

Function Parameters:

- *string* **$host** host server for the database
- *string* **$database** the database name to connect to
- *string* **$username** username to connect to the database with
- *string* **$password** the password for the database

connect

Connects to a database.

bool function Database::delete($table, $constraints) *[line 223]*

Function Parameters:

- *string* **$table** the table to delete from
- *assoc_array* **$constraints** the constriants on what to delete (where x = y)

delete

Delete a record from the database

bool function Database::disconnect() *[line 95]*

disconnect

Closes the connection to the active database

mixed function Database::escape($string) *[line 147]*

Function Parameters:

- *string* **$string** what we want to escape

escape

Smart escape string. Escapes data based on type.

bool function Database::free_result() *[line 121]*

free_result

Frees all allocated memory from the last query

object results function Database::get($table, [$constraints = null], [$columns = '*'], [$limit = null], [$orderBy = null], [$order = 'asc']) *[line 265]*

Function Parameters:

- *string* **$table** table to run the query on
- *assoc_array* **$constraints** constrains of the query
- *mixed* **$columns** the columns to select data on can be string or assoc_array
- *integer* **$limit** the number of rows to select
- *mixed* **$orderBy** the column to order the results by
- *string* **$order** asc or desc order

get

Generates and runs a select query, formats the results into an object along with the

number of rows, and clears all the resources from the query.

string function Database::get_error() *[line 134]*

get_error

Gets the mysql error and formats it into a string containing the error number and error description

bool function Database::insert($table, $data) *[line 168]*

Function Parameters:

- *string* **$table** the table to insert the record to
- *assoc_array* **$data** the data to insert, using an associative array

insert

Insert a record into the database

integer function Database::num_rows() *[line 245]*

num_rows

Gets the number of rows returned by the last run query

void function Database::optimize($table) *[line 352]*

Function Parameters:

- *string* **$table** the table to optimize (remove overhead from)

optimize

Optimizes a given table. (Removes a tables overhead caused by deletion of rows)

resource function Database::query($queryString) *[line 108]*

Function Parameters:

- *string* **$queryString** the sql query to run on the database

query

Run a query on the database

bool function Database::update($table, $constraints, $data) *[line 193]*

Function Parameters:

- *string* **$table** the table to update record(s) from
- *assoc_array* **$constraints** the constraints on what to update
- *assoc_array* **$data** the new data for the record(s)

update

Update record(s) in the database

Class glue

[line 50]

glue

Provides an easy way to map URLs to classes. URLs can be literal strings or regular expressions.

When the URLs are processed: * deliminators (/) are automatically escaped: (\/) * The beginning and end are anchored (^ $) * An optional end slash is added (/?) * The i option is added for case-insensitive searches

Example:

$urls = array('/' => 'index', '/page/(\d+) => 'page');

class page { function GET($matches) { echo "Your requested page " . $matches[1]; } }

glue::stick($urls);

- **Package** affero
- **Sub-Package** libraries

void function glue::stick($urls) *[line 63]*

Function Parameters:

- *array* **$urls** The regex-based url to class mapping

stick

the main static function of the glue class.

- **Static**
- **Throws** Exception Thrown if corresponding class is not found
- **Throws** Exception Thrown if no match is found
- **Throws** BadMethodCallException Thrown if a corresponding GET,POST is not found

Class Input

[line 24]

Input

This class takes all input, and passes it through some sanitization, and xss filtering before it is used.

- **Package** affero
- **Sub-Package** libraries

Constructor *void* function Input::__construct() *[line 33]*

__construct

This method will do all the bits that should be done as soon as the class is intiated. It unregisters globals and runs $_GET, $_POST, and $_COOKIE through the clean_input

method from the get go.

mixed function Input::clean_input($input) *[line 95]*
Function Parameters:

- *mixed* **$input** an array/object or string to clean

clean_input
removes magic quotes, standarizes new lines, and runs the xss filter on input

- **Access** public

string function Input::clean_key($key) *[line 137]*
Function Parameters:

- *string* **$key** the key to check

clean_key
A simple helper function that kills the application when it detects a nasty key in an array/object such as those containing non-alphanumeric text (with a few exceptions)

- **Access** public

mixed function Input::cookie([$key = '']) *[line 183]*
Function Parameters:

- *string* **$key** the key you want from $_COOKIE

cookie

an alias for $_COOKIE that has been filtered for xss

- **Access** public

void function Input::deep_unset($key, $value) *[line 69]*

Function Parameters:

- *string* **$key** the key for the global to unregister
- *string* **$value** the data in the global to check for arrays

deep_unset

A small helper function that recursively unsets a global.

- **Access** private

mixed function Input::fetch_global_data(&$global, $key) *[line 213]*

Function Parameters:

- *array* **&$global** the global to retrieve data from
- *string* **$key** the index (key) of the data to retrive

fetch_global_data

this simple method grabs the xss filtered data from the global variable passed to it.

- **Access** private

mixed function Input::get([$key = '']) *[line 156]*
Function Parameters:

- *string* **$key** the key you want from $_GET

get
an alias for $_GET that has been filtered for xss

- **Access** public

mixed function Input::post([$key = '']) *[line 170]*
Function Parameters:

- *string* **$key** the key you want from $_POST

post
an alias for $_POST that has been filtered for xss

- **Access** public

mixed function Input::server([$key = '']) *[line 197]*
Function Parameters:

- *string* **$key** the key you want from $_SERVER

server

an alias for $_SERVER that has been filtered for xss

- **Access** public

string function Input::xss_filter($data) *[line 241]*

Function Parameters:

- *string* **$data** the data to be filtered

xss_filter

This is a xss filter that should remove most nasty code entered into the site via form of url.

This function is not created by William Duyck. Please observe the legal rights of the following:

- **Author** Christian Stocker < christian.stocker@liip.ch>
- **Copyright** Copyright (c) 2001 - 2008 Liip AG
- **Link** http://svn.bitflux.ch/repos/public/popoon/trunk/classes/externalinput.php
- **Access** private
- **License** Apache License version 2.0

Class Rest

[line 22]

Rest

A collection of helper functions to make working with the api and extending it quicker and easier

- **Package** affero
- **Sub-Package** libraries

Rest::$alias

mixed = array(
'slug' => 'areaSlug',
'parent' => 'areaParentSlug',
'url' => 'areaURL',
'description' => 'areaDescription',
'name' => 'areaName',
'timeID' => 'timeRequirementID',
'timeShort' => 'timeRequirementShortDescription',
'timeLong' => 'timeRequirementLongDescription',
'date' => 'metricDate',
'qty' => 'metricQty',
'locale' => 'localeID',
'localeQty' => 'metricLocaleQty',
'tag' => array(
'slug' => 'skillTag',
'name' => 'skillName'
)) *[line 29]*

- **Access** protected

Rest::$data

mixed = *[line 27]*

- **Access** protected

Rest::$format

mixed = *[line 25]*

- **Access** protected

void function Rest::GET($args) *[line 57]*
Function Parameters:

- *array* **$args** the aguments supplied by gluephp about the url

GET

Responds to all get requests to the api and routes calls off to the relevant functions to produce a response. It is also in charge of formatting the response into either XML or JSON.

void function Rest::get_area($args) *[line 114]*
Function Parameters:

- *array* **$args** the aguments supplied by gluephp about the url

get_area

Spits out all the data relating to areas based on constraints passed via the url IF required.

- **Access** private

void function Rest::get_locale($args) *[line 418]*
Function Parameters:

- *array* **$args** the aguments supplied by gluephp about the url

get_locale

Gets the locale metric data and exposes it for the read only api

- **Access** private

void function Rest::get_metric($args) *[line 349]*
Function Parameters:

- *array* **$args** the aguments supplied by gluephp about the url

get_metric
Gets the metrics data and exposes it for the read only api

- **Access** private

void function Rest::get_time($args) *[line 281]*
Function Parameters:

- *array* **$args** the aguments supplied by gluephp about the url

get_time
Responds with data about time requirements based on constraints passed via the url IF required.

- **Access** private

void function Rest::respond($status, $data, $mime) *[line 99]*
Function Parameters:

- *int* **$status** the relevant http status code
- *string* **$data** the data to respond with
- *string* **$mime** the mime type of the content we are responding with

respond

This is used to send the response by setting the headers, and echoing the content to the calling body.

- **Access** private

Class Utility

[line 22]

Utility

Contains a collection of utility methods that are not specific to any area of the application.

- **Package** affero
- **Sub-Package** libraries

Constructor *void* function Utility::__construct() *[line 30]*

__construct

loads the affero configuration file and assigns config details to $this->config

bool function Utility::check_auth() *[line 126]*

check_auth

this is a simple helper function that checks to see if a users is or is not logged in. If they are not it redirects them to the login page and returns false, else it returns true with no redirect.

- **Access** private

string function Utility::current_url() *[line 46]*

current_url

This function get the url of the current page the user is viewing, and sticks all the bits together as PHP does not do this for you.

string function Utility::hash_string($string, [$salt = null]) *[line 111]*

Function Parameters:

- *string* **$string** the string to be hashed
- *string* **$salt** the salt to use when hashing the $string

hash_string

This function take string and optional salt to create a sha512 hash that is then returned.

If a salt is not provied will just hash {string} else will hash {string}_{salt}

string function Utility::site_url([$localtion = '']) *[line 77]*

Function Parameters:

- *string* **$localtion** the page/file on site to point to

site_url

This function generates a url that points to a page on the same site based on the domain provided in the configuration file.

bool function Utility::valid_email($email) *[line 93]*

Function Parameters:

- *string* **$email** The email to check

valid_email

This function returns true or false depending on whether the input it receives is a valid url

- **TODO** valid_email
  - Grab regex and return bool based on preg_match of said regex

Class View

[line 27]

View

A collection of helper functions for use in views to make life a little easier and development quicker

- **Package** affero
- **Sub-Package** libraries

void function View::head([$title = 'untitled']) *[line 88]*

Function Parameters:

- *string* **$title** The title of the page to display in the browser tab/title bar

head

This creates and prints out the default html for the <head> tags, for all views.

void function View::load($name, [$data = null], $view) *[line 39]*

Function Parameters:

- *string* **$view** the name of the view file to load
- *assoc_array* **$data** the data to pass to the view file
- **$name**

load

This function gets and loads a view file and passes it data it receives via an associative array, and makes it available to the view.

void function View::navigation() *[line 109]*

navigation

This function checks if a user is logged in and presents a different set of navigation options if so.

It prints the navigation as html

void function View::script($name) *[line 146]*

Function Parameters:

- *string* **$name** the name of the js file (ex. '.js')

script

This simple helper prints the html for loading scripts within the affero js directory

void function View::stylesheet($name) *[line 133]*

Function Parameters:

- *string* **$name** the name of the css file (ex. '.css')

stylesheet

This simple helper prints the html for loading stylesheets within the affero css directory

index.php

All Tests

This file gets, and runs all the testcases available for affero. All tests are written using the simpletest framework, as is this file. This file acts like a test suite.

- **Package** affero
- **Sub-Package** testing
- **Author** William Duyck
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

require_once **'/simpletest/autorun.php'** *[line 19]*

configuration_file.php

Configuration Test Case

This file contains all the code used to test the configuration file that sets the application constants for affero

- **Package** affero
- **Sub-Package** testing
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

require_once **dirname(__FILE__).'/../../../app/config.php'** *[line 18]*

require_once **'/simpletest/autorun.php'** *[line 17]*

database_abstraction_layer.php

Database Abstraction Layer Test Case

This file contains all the code used to test the database abstraction layer. Tests are run automatically on file load.

- **Package** affero
- **Sub-Package** testing
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** MPL 1.1/LGPL 2.1/GPL 2.0 William Duyck

require_once **dirname(__FILE__).'/../../../app/libraries/database.lib.php'** *[line 19]*

require_once **'/simpletest/autorun.php'** *[line 17]*

user_management.php

Test Of User Management Class

This testcase will check that the User Management Class is fully functional and fit for purpose.

- **Package** affero
- **Sub-Package** testing
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** 2011 William Duyck
- **License** MPL

require_once **dirname(__FILE__).'/../../../app/modules/user.php'** *[line 24]*

require_once **dirname(__FILE__).'/../../../app/libraries/database.lib.php'** *[line 22]*

require_once **dirname(__FILE__).'/../../../app/libraries/input.lib.php'** *[line 21]*

require_once **dirname(__FILE__).'/../../../app/libraries/utilities.lib.php'** *[line 20]*

require_once **'/simpletest/autorun.php'** *[line 18]*

utility.php

Utilities Test Case

This file contains all the code used to test the utilities class that contains general functions that can be of use in areas of the application but that are not specific to a particular part of the application.

- **Package** affero
- **Sub-Package** testing
- **Author** William Duyck < wduyck@gmail.com>
- **Version** 0.1
- **Copyright** MPL 1.1/LGPL 2.1/GPL 2.0 William Duyck

require_once **dirname(__FILE__).'/../../../app/libraries/utility.lib.php'** *[line 20]*

require_once **'/simpletest/autorun.php'** *[line 18]*

Class AllTests

[line 27]

AllTests

This test suite runs all the tests it finds in the same directory as itself.

- **Package** affero
- **Sub-Package** testing

Constructor *void* function AllTests::AllTests() *[line 40]*

AllTests

The constructor for the class. It will run automatically when the class is run, and in turn, run all the test cases in the current directory.

It scans the /test/php/testcase directory for all files and attempts to run them as testcases. It then gets the results and reports any errors.

Class TestConfiguration

[line 20]

- **Package** affero
- **Sub-Package** testing

void function TestConfiguration::testConfigObjectsExist() *[line 26]*

test configuration constants exist and setup

Class TestOfDatabaseAbstractionLayer

[line 30]

TestOfDatabaseAbstractionLayer

This class tests all methods in the Database class. It is an extension of the UnitTestCase class provided by the simpletest unit testing framework.

All database connections will be made to the database at 'localhost' named 'ccw_dev' with the username 'root'. The database is set to have no password.

- **Package** affero
- **Sub-Package** testing

void function TestOfDatabaseAbstractionLayer::setUp() *[line 37]*

setUp

Setup for a new database object for each test

void function TestOfDatabaseAbstractionLayer::tearDown() *[line 48]*

tearDown

Remove the database object at the end of each test to avoid tests influencing the result of eachother

void function TestOfDatabaseAbstractionLayer::testDatabaseConnectDisconnect() *[line 64]*

testDatabaseConnectDisconnect

Test if able to connect/disconnect to the database. Both methods should assert true if no errors were encountered.

The connect method should accept a host, database name, username, and password argument

The disconnect method should not accept any arguments

void function TestOfDatabaseAbstractionLayer::testDatabaseDeleteMethod() *[line 235]*

Test that we can delete from the database by removing the test data we

just added in the insert method test

void function TestOfDatabaseAbstractionLayer::testDatabaseFreeResultMethod() *[line 275]*

test that the memory from the query is freed successfully. Will assert true on success.

void function TestOfDatabaseAbstractionLayer::testDatabaseGetErrorMethod() *[line 293]*

test that error reporting works by attempting to select from a

non-existant table in the database

void function TestOfDatabaseAbstractionLayer::testDatabaseGetMethodWithLimitations() *[line 312]*

Test by attempting to retrieve just the email of user wduyck in the database and that the method returns an object representation of the data it gets.

There should be one row, and the email should be wduyck@gmail.com

void function TestOfDatabaseAbstractionLayer::testDatabaseInitConnectAlias() *[line 80]*

testDatabaseInitConnectAlias

Test the Database class connects to a database on initilization provided information of the database to connect to.

The alias should be used in the following manner:

$database = new Database(host, database name, username, password);

void function TestOfDatabaseAbstractionLayer::testDatabaseInsertMethod() *[line 172]*

testDatabaseInsertMethod

This test will create a new user in the database and then check that we can get the same information back using the query method afterwards.

It works on the assumption that the query method has passed its test and works as expected.

Test data:

- username = test_user
- password = sha512 hash of 'eiugbkwabgp9bgv3eBÂ£Egib'
- email = test@example.com

The insert method should accept the following arguments * table name * an associative array of data in the form 'field name' => 'value'

void function TestOfDatabaseAbstractionLayer::testDatabaseNumRowsMethod() *[line 256]*

Test the number of rows returned by a query by getting all the rows from the user table and running the num_rows method on the returned results, then compare this to the number returned by mysql_num_rows.

void function TestOfDatabaseAbstractionLayer::testDatabaseQueryMethod() *[line 111]*

testDatabaseQueryMethod

This test will attempt to get the first user in the database and check that the details match with what should already be in user record one.

Test data stored:

- username = john.doe
- password = 62dee3a48d7a3d9e190c8ba3f2ada76c1bb85aa41b5d5f8f7ed82fef028aaaaa305231c82275b4cd2a4acac5119c6f5ab03666fea9080b776418001d5c2ab206
- email = john.doe@example.com

The query method should accept JUST a sql query string.

void function TestOfDatabaseAbstractionLayer::testDatabaseTableOptimizeMethod() *[line 332]*

Test attempts to optimise the user table, should assert true on success

void function TestOfDatabaseAbstractionLayer::testDatabaseUpdateMethod() *[line 213]*

testDatabaseUpdateMethod

This test will check to see if we can update information in the database by modifying the test data added in testDatabaseInsertMethod.

It will attempt to change the email to 'test@test.com' and will then use the query method to check that the data has been changed.

The update method should accept the following arguments: * table name * associative array to select the row(s) to modify in the form 'field' => 'value' * associative array of new data in the form 'field' => 'value'

void function TestOfDatabaseAbstractionLayer::testEscapeMethod() *[line 143]*

testEscapeMethod

This test will pass two strings into the function and check that the output is what is expected which is a string with quotation marks around it. Should the string already contain quotation marks then these should be escaped with a backslash.

An interger and boolean will also be passed into the method to check that it spits them straight back out unchanged.

The escape method should accept anything that is passed into the first argument.

Class TestOfUserManagementClass

[line 26]

- **Package** affero
- **Sub-Package** testing

void function TestOfUserManagementClass::setUp() *[line 33]*

setUp

create an new instance of the user management class

void function TestOfUserManagementClass::tearDown() *[line 44]*

tearDown

destroies the instance of the user management class ready for the next test

void function TestOfUserManagementClass::testSessionCreateAndDestroy() *[line 62]*

TestSessionCreate

This function will test that a new session is created on successfull login and that the session is destroied on logout

Will attempt to login with a number of invlaid credentials, then use valid credentials to complete a successful login. The login method from the user class should assert true on success and false on failure

The function will then test to see if a session has been created for the login and that it stores the correct username.

Class TestOfUtilitiesClass

[line 29]

TestOfUtilitiesClass

This class will test the utilities class. It will test all the methods that are available for use from the utilities class. These methods are not related to any specifc areas of the application.

- **Package** affero
- **Sub-Package** testing

void function TestOfUtilitiesClass::setUp() *[line 36]*

setUp

Setup for a new utilities object for each test

void function TestOfUtilitiesClass::tearDown() *[line 47]*

tearDown

Remove the utilities object at the end of each test to avoid tests influencing the result of eachother

void function TestOfUtilitiesClass::testCurrentUrlMethod() *[line 64]*

testCurrentUrlMethod

This test will attempt to check that the url returned by the current_url method is the same as the url of this file.

The current_url method should not accept any parameters and should return a string that matches the url of the current page.

This test will only pass when the expected url of this test file is known and manually programed into the regex of the assertion.

void function TestOfUtilitiesClass::testHashStringMethod() *[line 91]*

testHashStringMethod

This test will attempt to verify that the hash_string method returns an sha512 hash that has a salt.

The hash_string method should accept a string to hash and an optional salt.

Test Data:

- string to hash = 'HashMePlease'
- salt = 'WithASaltToo'

Algorithm:

if salt provided the method should hash {string}_{salt} else just {string}

Expected Returns:

- without salt = '632c18211a33f50e32981cc46c95d015098bcfa5432f3d09e0fadf1effb2189fd208b4e8828071011fe187a592c53f0ccf95b4ff97b5054fba0ac87a6eedc5ca'
- with salt = 'babf3f6d79803c7cb6d07737dbfeb0a9c3cbe81411be23c60549dd3155a1288b96b650d32331e7d69f0690c87c6687cc68d1fb01493f14c7d8678d3d62d83491'

Appendices

Appendix A - Class Trees

Package affero

AllTests

- TestSuite
  - AllTests

Controller

- Controller
  - Affero
  - Dashboard
  - Manage
  - Rest
  - User

Database

- Database

glue

- glue

Input

- Input

TestConfiguration

- UnitTestCase
  - TestConfiguration

TestOfDatabaseAbstractionLayer

- UnitTestCase
  - TestOfDatabaseAbstractionLayer

TestOfUserManagementClass

- UnitTestCase
  - TestOfUserManagementClass

TestOfUtilitiesClass

- UnitTestCase
  - TestOfUtilitiesClass

Utility

- Utility
  - View

Package default

Appendix D - Todo List

In Package affero

In **Database**:

- DAL Rewrite
  - Rewite all methods to use the newer mysqli functions over the mysql ones

In **User::invite()**:

- invite system
  - refactor the code

In **Manage**:

- Rethink edit functions
  - remove area_edit
  - remove area_edit view
  - refactor area to control add/edit/delete all in one
  - refactor area_add to work for both add and edit

In **User**:

- user class items
  - create add user function

In **Utility::valid_email()**

- valid_email
  - Grab regex and return bool based on preg_match of said regex

Index

A

C

D

G

I

M

R

T

U

V

_