

People_scope API Manual



Contents

question.class.php	0
database.class.php	0
email.class.php	0
errorhandler.class.php	0
form.class.php	0
table.class.php	0
template.class.php	0
tools.function.php	0
Function convertDecimalToMinutes	0
Function convertMinutes2Hours	0
Function convertUIdate	0
Function createDateField	0
Function databaseToUI	0
Function fileExt	0
Function formatArray	0
Function formatDateResponce	0
Function formatDateUI	0
Function parseArray	0
Function pp	0
Function showTrace	0
Function showVars	0
Function stripElements	0
Function trace	0
Global Variable \$GLOBALS['style']	0
Global Variable \$GLOBALS['trace']	0
Class CustomException	0
Method logError	0
Method queryError	0
Class db	0
Var \$dbh	0
Var \$dsn	0
Var \$lastQuery	0
Constructor construct	0
Method delete	0
Method getDNSString	0
Method insert	0
Method prepareToQuery	0
Method query	0
Method select	0
Method update	0
Class email	0
Var \$CrLf	0
Var \$header	0
Var \$mail	0

Var \$mime	0
Var \$template	0
Constructor email	0
Method append_file	0
Method append_html	0
Method append_text	0
Method recordEmail	0
Method send	0
Method sender	0
Method setHeader	0
Method setHTMLtemplate	0
Method setTXTtemplate	0
Method subject	0
Method construct	0
Class form	0
Var \$autoRefresh	0
Var \$autoRefreshcheckboxs	0
Var \$autoRefreshInputEnter	0
Var \$db	0
Var \$enctype	0
Var \$fck_configUrl	0
Var \$fck_height	0
Var \$fck_width	0
Var \$form	0
Var \$formScript	0
Constructor construct	0
Constructor form	0
Method draw	0
Method formFooter	0
Method formHeader	0
Method formScript	0
Method freeFormSubmit	0
Method inputfield	0
Method setAutoRefresh	0
Method SetFckConfigUrl	0
Method unsetAutoRefresh	0
Class table	0
Var \$basePage	0
Var \$columnsWidth	0
Var \$filterArray	0
Var \$footer	0
Var \$headerArray	0
Var \$id	0
Var \$identifier	0
Var \$link_action	0
Var \$link_field	0
Var \$name	0
Var \$nolink	0
Var \$removeColumnArray	0

Var \$row class field name	0
Var \$row class name	0
Constructor construct	0
Method buildTd	0
Method buildWhereArrayFromRequest	0
Method generateDisplayTable	0
Method getFilterType	0
Method getOrderType	0
Method removeColumn	0
Method setBasePage	0
Method setColumnsWidth	0
Method setFilter	0
Method setFooter	0
Method setHeader	0
Method setIdentifier	0
Method setIdentifierPage	0
Method setLinkAction	0
Method setLinkField	0
Method setPrimaryId	0
Method setRowClassFieldName	0
Method setRowClassName	0
Method setTableName	0
Method destruct	0
Class template	0
Var \$assigned	0
Var \$repeatRegion	0
Var \$replacer	0
Var \$template file	0
Constructor construct	0
Constructor template	0
Method assign	0
Method assignArray	0
Method assignBlank	0
Method assignRepeat	0
Method BuildAssigned	0
Method display	0
Method fetch	0
Method replace	0
Method set	0
Appendix A - Class Trees	0
PeopleScope	0
Appendix D - Todo List	0

question.class.php

- **Package** PeopleScope
- **Filesource** [Source Code for this file](#)

database.class.php

Database Class,

This class is Database abstraction layer

Example:

```
define('DB_USER','root');
define('DB_PASS','password');
define('DB_HOST','localhost');
define('DB_DBASE','test_db');
define('DB_TYPE','mysql');

try{$db = new db();}
catch(CustomException e){ echo $e->logError(); }

try{$result = $db->select('select * from test_table');}
catch(CustomException e){echo $e->queryError();}

foreach($result AS $row){
    echo 'col1 =' . $row['col1'];
    echo 'col2 =' . $row['col2'];
    echo 'col3 =' . $row['col3'];
}
```

- **Package** PeopleScope
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

require_once ['errorhandler.class.php'](#) *line 40*

required error handler

require_once ['tools.function.php'](#) *line 36*

required general tool

email.class.php

Email Class,

This class is used create and send email, can also record emails in a db, and can use the template class to format Html

- **Package** PeopleScope
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

`require_once 'Mail/mime.php'` [*line* [20](#)]

Pear Mime class

`require_once 'Mail.php'` [*line* [16](#)]

Pear Mail class

errorhandler.class.php

CustomException Class,
Generates a custom exception

Example:

throw an exception

```
if(empty($value)){  
    throw new CustomException('value can not be empty');  
}
```

try/catch

```
try{  
    $db->query('SEELCT * FROM notable ');  
}catch(CustomException $e){  
    echo $e->logError();  
}
```

- **Package** PeopleScope
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

form.class.php

Form Class,

This class is used to take a input String in a standardised format and convert to a Html Form

Example:

```
$formVars = "  
Title*:1:select:Ms|Ms,Mrs|Mrs,Miss|Miss,Mr|Mr::required:notype  
Home Phone:5:editor::400|400::optional:notype  
First Name*:2:text::::required:notype  
Last Name*:3:text::::required:notype  
Work Phone:4:text::::optional:notype  
Home Phone:5:text::::optional:notype  
Mobile Phone*:6:text::::required:notype  
Email*:7:text::::required:notype  
Address:8:textarea::::optional:notype  
Suburb:9:text::::optional:notype";
```

```
$form = new form($formVars);  
$formVal = $form->formHeader("");  
$formVal .= $form->draw();  
$formVal .= $form->freeFormSubmit();  
$formVal .= $form->formFooter();
```

```
echo $formVal
```

- **Package** PeopleScope
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

table.class.php

Table Class,

This class is used to take the output from a select Query and turn it into a filterable table

That class ajax from JQuery script js/table-layout.js to handle displaying of filtered information

- **Package** PeopleScope
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

template.class.php

Template Class,

This class is used to take a input to generate templates

Example:

```
$template = new template('template/index.html');

$template->assign('var1', 'Employment list');
                $template->assignArray(array{'var2'=>'John',           'var3'=>'mary',
'var4'=>'<strong>frank</strong>'});

$ArrayVars[0]['name'] = 'john';
$ArrayVars[0]['age'] = '14';
$ArrayVars[1]['name'] = 'mary';
$ArrayVars[1]['age'] = '42';
$ArrayVars[2]['name'] = 'frank';
$ArrayVars[2]['age'] = '98';

$template->assignRepeat('agelist', $ArrayVars);
$template->replace('..', '../');

$template->display();
```

- **Package** PeopleScope
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

require_once '[errorhandler.class.php](#)' *line 35*

required error handler

tools.function.php

Tools Function,

At set of general tool to help with development

- **Package** PeopleScope
- **Sub-Package** Base
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

Integer function convertDecimalToMinutes(\$percent) [*line* [356](#)]

Function Parameters:

- *Integer* **\$percent**

Will take in a percent amount and return values in mins

eg. 50% = 30mins

String function convertMinutes2Hours(\$Minutes) [*line* [371](#)]

Function Parameters:

- *Integer* **\$Minutes**

Will take in a amount in minutes and return the value in the amount of hours : minute

124 minutes = 2:04

void function convertUIdate(\$start, \$end, &\$ARRAY) [*line* [283](#)]

Function Parameters:

- *String* **\$start** start date

- *String* **\$end** end date
- *Array* **&\$ARRAY** returns by reference

Will convert the output from a JQuery UI date field format
format must be in Au date format

- **Link** <http://docs.jquery.com/UI/Datepicker>
- **Link** <http://docs.jquery.com/UI/Datepicker/formatDate>
- **TODO** remember how this works :S

an function createDateField(\$date, [\$yearRange = 10], [\$startyear = NULL], \$name) [line [179](#)]

Function Parameters:

- *String* **\$name** Name used for selection box
- *String* **\$date** Date That should be selected
- *Integer* **\$yearRange** How many years should be displayed
- *Integer* **\$startyear** What year should the list start at

This will take in a name for the field types and a date that you wish and format an option list for select boxes, and return them in an associated array for year, month and day

String function databaseToUI(\$dbdate) [line [325](#)]

Function Parameters:

- *String* **\$dbdate** database date format yyyy-mm-dd

converts a database date value and conversts to Au date format
2001-04-02 = 02/04/2001

String function fileExt(\$type) [line [341](#)]

Function Parameters:

- String **\$type** Mime type

return file extention based on Mime type for common doc types doc, docx, pdf

void function formatArray(\$array, \$name) [line [120](#)]

Function Parameters:

- Array **\$array** Array to read
- String **\$name** name you want to give the array

Will input an associate array and output in a readable format

String function formatDateResponse(\$name, [&\$listArray = NULL]) [line [241](#)]

Function Parameters:

- String **\$name** the name of the fields we are looking for
- Array **&\$listArray** alternate array to look at, \$_REQUEST by default

This will take in an Array or if no array give will default to the \$_REQUEST

global looking for associated name type and remove the elements and replace the with a \$name field with a value of the compiled date, if none found then will put in today's date associated name types :

\$name.'__day'

\$name.'__month'

\$name.'__year'

\$name = \$name.'__year-'\$name.'__month-'\$name.'__day 00::00::00'

void function formatDateUI(\$date) [line [219](#)]

Function Parameters:

- String **\$date** database date format

Will take a Au date from jQuery Datepicker and convert to a database date format

- Link <http://docs.jquery.com/UI/Datepicker/formatDate>
- Link <http://docs.jquery.com/UI/Datepicker>

Array function `parseArray($string, $beg_tag, $close_tag, [$remove_tag = true])` [line [144](#)]

Function Parameters:

- *String* **\$string** String to parse
- *String* **\$beg_tag** start delimiter
- *String* **\$close_tag** end delimiter
- *Bool* **\$remove_tag** true return content without delimiter

Will take in a **String** and a start and end delimiter and will return the content between the delimiter

void function `pp($var, [$tracer = false])` [line [38](#)]

Function Parameters:

- *Mixed* **\$var** Value that will be echoed out
- *bool* **\$tracer** true display to screen, false store in global \$trace return void

A debugging tool

This will produce a error message to the screen displaying the value enter it will take in vars | arrays | object it will show the page | Line | function that the echo accured and can be turned off thought the constant `TURN_ON_PP`

void function `showTrace()` [line [83](#)]

Output `$GLOBAL['trace']` Trace

String function showVars() [line 93]

Will return a html form with all the results of the \$GLOBAL vars

\$_POST, \$_GET, \$_COOKIE, \$_SESSION, \$trace

void function stripElements(\$elements, &\$array, \$array) [line 161]

Function Parameters:

- **String \$elements** Comma separated list of names
- **Array \$array** The array to remove them from via reference
- **&\$array**

Will take in a list of associated array names and remove from that array via reference

void function trace(\$var, [\$newline = true]) [line 68]

Function Parameters:

- **Mixed \$var** Value that will be echoed out
- **Bool \$newline** If type strip tags from string

Debugging tool that that will store the outcomes into a \$GLOBAL['trace'] var

- **See showTrace for output**

\$GLOBALS['style']

string = "style="font-family: Arial, Helvetica, sans-serif; font-size: 11px; padding:10px; background-color: #ffcccc; width:100%; border:solid 1px #000" [line 19]

Global style for error messages

\$GLOBALS['trace']

string = array() [line 24]

Class CustomException

[line [34](#)]

Generates a custom exception

- **Package** PeopleScope
- **Sub-Package** Base

string function CustomException::logError() [line [40](#)]

Generate a formatted excption error

- **Access** public

void function CustomException::queryError(\$query) [line [63](#)]

Function Parameters:

- *\$query* **\$query** Sql query that was being used at the time of execution

Will append the query being used to the begining of a logError output

This is used to create an exception error for query execution, to produce error that also have the query displayed with in the error output

- **Access** public

Class db

[[line 49](#)]

This class is Database abstraction layer

- **Package** PeopleScope
- **Sub-Package** Base

db::\$dbh

mixed = [[line 52](#)]

db::\$dsn

mixed = [[line 51](#)]

db::\$lastQuery

mixed = [[line 53](#)]

Constructor *Void* function db::__construct() [[line 61](#)]

constructor to initiate database conection

Array function db::delete(\$sql) [[line 125](#)]

Function Parameters:

- *string* \$sql

Will instigate a DELETE query and return true if no problems;

- **Access** public

String function db::getDNSString() [[line 80](#)]

returns current DSN string used to connect ot the server

- **Access** public

Array function db::insert(\$sql) [[line 108](#)]

Function Parameters:

- *String* **\$sql**

Will instigate a INSERT query and return the inserts autocomplete Id;

- **Access** public

string function db::prepareToQuery(\$query, \$params) [[line 190](#)]

Function Parameters:

- *string* **\$query** Query template
- *array* **\$params** Array of inputs

Merge query template with array

Query template and array set merger function Will taken in the Query string template and the array of query elements and combine the to show the fully converted string used in the prepare Note: only use as example for debugging purposes

- **Access** public

array function db::query(\$sql) [[line 160](#)]

Function Parameters:

- *string* **\$sql**

Will instigate a query and return a recordset;

- **Access** public

Array function db::select(\$sql) [[line 90](#)]

Function Parameters:

- *String* **\$sql** select sql string to be executed

Will instigate a SELECT query and return an array of responses

- **Access** public

array function db::update(\$sql) [[line 142](#)]

Function Parameters:

- *string* **\$sql**

Will instigate a UPDATE query and return true if no problems;

- **Access** public

Class email

[line [28](#)]

This class is used create and send email

- **Package** PeopleScope
- **Sub-Package** Base

email::\$crlf

String = "\n" [line [34](#)]

Carrage return

email::\$header

Araay = [line [58](#)]

Mail Header information

email::\$mail

Object = [line [46](#)]

Mail Object

email::\$mime

unknown_type = [line [40](#)]

Set Mime Type

email::\$template

Object = [line [52](#)]

Template Object

Constructor *void* function email::email() [line [73](#)]

Constructor for Email php4

void function email::append_file(\$file, \$type) [line [102](#)]

Function Parameters:

- *String* **\$file** file path to the, or file content
- *String* **\$type** Mime type of the thaile attached

converts the TEXT version of the email to be sent to be appended to the email

void function email::append_html(\$html) [line [83](#)]

Function Parameters:

- *String* **\$html**

converts the HTML version of the email to be sent to be appended to the email

void function email::append_text(\$text) [line [92](#)]

Function Parameters:

- *String* **\$text**

converts the TEXT version of the email to be sent to be appended to the email

void function email::recordEmail(\$table, \$content, [\$form_name = NULL]) [line [195](#)]

Function Parameters:

- *String* **\$table** table to store the email
- *String* **\$content** content of the email

- *String* **\$form_name** identifier of record, Default URI sent from

Record email into the database

void function email::send(\$to) [[line 231](#)]

Function Parameters:

- *String* **\$to** Email address to send too;

compile and Send email

void function email::sender(\$from) [[line 159](#)]

Function Parameters:

- *String* **\$from** email of the sender

Set the Sender info for header

void function email::setHeader(\$header) [[line 177](#)]

Function Parameters:

- *Array* **\$header** header param to header value

Set all other Header information as required

void function email::setHTMLtemplate(\$template, \$content) [[line 113](#)]

Function Parameters:

- *String* **\$template** path to template
- *String* **\$content** array of assigned variable conversion

Create a HTML email using the base template class

- **See** `template::assignArray`

`void function email::setTXTtemplate($table, $content)` [[line 128](#)]

Function Parameters:

- *String* **\$table** Form Class standardised format String
- *String* **\$content** Content

Will create and output from Form Class standardised format String To create a TEXT email in a reable format

- **TODO** confirm this is correct

`void function email::subject($subject)` [[line 167](#)]

Function Parameters:

- *String* **\$subject**

Set the Subject line fo the email

`void function email::_construct($type)` [[line 65](#)]

Function Parameters:

- *unknown_type* **\$type**

Constructor for Email php5

Class form

[line 42]

This class is used to take a input String in a standardised format and convert to a Html Form

- **Package** PeopleScope
- **Sub-Package** Base

form::\$autoRefresh

String = [line 87]

JavaScript handle onChange for Form input types command

- **TODO** should be removed for JQuery ready({})

form::\$autoRefreshcheckboxs

String = [line 95]

JavaScript handle onClick for Form input types command

use this for check boxes and radio

- **TODO** should be removed for JQuery ready({})

form::\$autoRefreshInputEnter

String = [line 103]

JavaScript handle onkeyup for Form input types command

user this to to action on an text input with an enter button

- **TODO** should be removed for JQuery ready({})

form::\$db

Object = [line [48](#)]

Database object

form::\$enctype

String = [line [60](#)]

Form enctype type

form::\$fck_configUrl

String = [line [109](#)]

Url Tor find fck config file

form::\$fck_height

Integer = 500 [line [78](#)]

Height size of fck editor

form::\$fck_width

Integer = 300 [line [72](#)]

Width size of fck editor

form::\$form

String = "" [line [54](#)]

Holds string value for form builder

form::\$formScript

String = [line [66](#)]

Java scripts reloaded to this class

Constructor *void* function form::__construct(\$form) [line [117](#)]

Function Parameters:

- *String* **\$form** input values for form information

Consrtructor

Constructor *void* function form::form(\$form) [line [127](#)]

Function Parameters:

- *String* **\$form** input values for form information

Consrtructor php4

String function form::draw([\$column = NULL]) [line [212](#)]

Function Parameters:

- *Integer* **\$column** Define which column shold be returned

Will Generated the required input fields from the \$this->form information information can be broken up using the string '-----'

String function form::formFooter() [line [198](#)]

Setup closing form tag

string function form::formHeader(\$action, [\$method = NULL], [\$formname = NULL]) [line [163](#)]

Function Parameters:

- *String* **\$action** Form action parameter
- *String* **\$method** Form Method parameter

- *String* **\$formname** Form Name and Id parameter

Setup the form tag ready for the inputs

void function form::formScript(\$script) [*line* [142](#)]

Function Parameters:

- *String* **\$script** Javascript command or function

Set onSubmit Form scripts

- **TODO** should be removed for JQuery ready({})

void function form::freeFormSubmit([\$value = 'Submit']) [*line* [151](#)]

Function Parameters:

- *String* **\$value** Button lable/Name

Standards Submit Button

Html function form::inputfield(\$input) [*line* [254](#)]

Function Parameters:

- *String* **\$input** string eg.
State:10:select:ACT|ACT,NSW|NSW,VIC|VIC,QLD|QLD,SA|SA,NT|NT,WA|WA,TAS|TAS::optional:notype

Takes in a standard single row from \$this->form information and generated and html form input string

void function form::setAutoRefresh(\$command) [*line* [334](#)]

Function Parameters:

- *String* **\$command** Javascript code or function

Set the AutoRefresh function command

- **TODO** should be removed for JQuery ready({})

void function form::SetFckConfigUrl(\$url) [[line 243](#)]

Function Parameters:

- *String* **\$url** Url to new config file

Set a alternate FCK config file path

void function form::unsetAutoRefresh() [[line 346](#)]

Remove the AutoRefresh function command

- **TODO** should be removed for JQuery ready({})

Class table

[[line 21](#)]

This class is used to take the output from a select Query and turn it into a filterable table

- **Package** PeopleScope
- **Sub-Package** Base

table::\$basePage

unknown_type = NULL [[line 69](#)]

name of the file that is link to from identifier

- **Access** private

table::\$columnsWidth

Array = array() [[line 51](#)]

An Associated array of Key: fields names, Value: column with

- **Access** private

table::\$filterArray

Array = array() [[line 39](#)]

An Associated array of Key: fields names, Value: Filter type

- **Access** private

table::\$footer

String = [[line 100](#)]

Define a footer row for your table

- **Access** private

table::\$headerArray

Array = array() [[line 33](#)]

An Associated array of Key: fields names, Value: new column name

- **Access** private

table::\$id

String = [[line 57](#)]

String \$id name of table

- **Access** private

table::\$identifier

String = [[line 27](#)]

name of field that define the unique identifier for the table

- **Var** Id field
- **Access** private

table::\$link_action

String = 'show' [[line 88](#)]

Define &action type for page if identifier is set

- **Access** private

table::\$link_field

String = [[line 94](#)]

Define &action based on a fields name for page if identifier is set

- **Access** private

table::\$name

String = 'list' [[line 63](#)]

Name of table, used for id field

- **Access** private

table::\$nolink

Bool = false [[line 107](#)]

define if table links to another page for each row

- **Access** private

table::\$removeColumnArray

Array = array() [*line* [45](#)]

A list of field names in an arrays to be removed from the table columns

- **Access** private

table::\$row_class_field_name

String = [*line* [82](#)]

Define a class for a column

- **Access** private

table::\$row_class_name

String = [*line* [75](#)]

defined class name for the all the rows

- **Access** private

Constructor *void* function table::__construct([*\$id* = NULL]) [*line* [113](#)]

Function Parameters:

- *String* ***\$id*** name of table

Constructor for this table class

- **Access** public

void function table::buildTd(\$key, \$value) [*line* [516](#)]

Function Parameters:

- *String* **\$key**
- *String* **\$value**

This function will take the field values and wrap a td tag around it, It will also format the field based on the filter type

- **Access** private

String function table::buildWhereArrayFromRequest([\$full_like = NULL]) [*line* [548](#)]

Function Parameters:

- *Bool* **\$full_like** Will set all TEXT filter to %like% Default is like%

This function will take in the information from the \$_REQUEST global vars from a filter and conver it to a WHERE statement to be append to a query to filter the output

- **Access** public

`void` function table::generateDisplayTable(\$content) [[line 300](#)]

Function Parameters:

- *Array* **\$content**

This function will take the output of database query and generate a table for it, including headers and filters

- **Access** public

`string` function table::getFilterType(\$type, \$key, [\$content = NULL]) [[line 458](#)]

Function Parameters:

- *String* **\$type**
- *String* **\$key** fields name for a column
- *Array* **\$content** An array of field information for COMPILE type

This function will create a filter for a field based on its type

Filter types.

TEXT : A text box, entered text will do a like% search on the feild

VALUE : A select box and a text box, select show =,<=, >=, <> , text box hold the value

COMPILED : A Selct box showing all the possible entries in the Field

MONEY : see VALUE

- **TODO** Add <> to VALUE list
- **Access** private

`void` function table::getOrderType(\$type, \$key) [[line 502](#)]

Function Parameters:

- *String* **\$type**
- *String* **\$key**

Will wrap div information around the header name so we can add id for Asc Desc triggers for ordering

- **Access** private

void function table::removeColumn(\$columnArray) [*line* [184](#)]

Function Parameters:

- *Array* **\$columnArray**

An array of fields that should not be shown in the table

- **Access** public

void function table::setBasePage(\$basepage) [*line* [263](#)]

Function Parameters:

- *unknown_type* **\$basepage**

if using identifier then define the page the it links to Default is the current page

- **Access** public

void function table::setWidth(\$columnWidthArray) [*line* [194](#)]

Function Parameters:

- *Array* \$columnWidthArray

Associated array of Name of field for keys and the value is width of the field HTML rules apply, width in pixels '24', width percent '50%', width flexible '*'

- **Access** public

void function table::setFilter(\$filterArray) [*line* [175](#)]

Function Parameters:

- *Array* \$filterArray

Will using an input of an associated array add apply a filter type to a field name

eg.

\$array['name'] = 'TEXT';

\$array['age'] = 'VALUE';

Filter types.

TEXT : A text box, entered text will do a like% search on the field

VALUE : A select box and a text box, select show =,<=, >=, <> , text box hold the value

COMPILED : A Select box showing all the possible entries in the Field

MONEY : see VALUE

- **Access** public

void function table::setFooter(\$field) [*line* [151](#)]

Function Parameters:

- *String* **\$field**

Set the table Footer row for the table Html Footer rules apply

- **Access** public

void function table::setHeader(\$headerArray) [*line* [141](#)]

Function Parameters:

- *Array* **\$headerArray**

Will using an input of an associated array to rewrite field names to more readable header, if no match is found the header will be the field name

An Associated array with the key the name of the field and the value the name to change too

eg.

`$array['col1'] = 'First Column';`

`$array['col2'] = 'Second Column';`

- **Access** public

void function table::setIdentifier(\$field, [\$no_link = false]) [*line* [209](#)]

Function Parameters:

- *String* **\$field**
- *Bool* **\$no_link**

The name of the field that should be used to identify the row

if used will create a link on each row that would like to the page itself with a reference id to this field

```
onclick=\"location.href='index.php?action=show&id={id}'\" ";
```

- **Access** public

void function table::setIdentifierPage(\$page) [*line* [224](#)]

Function Parameters:

- *String* \$page

A page that the setIdentifier should goto if selected

- See [table::setIdentifier\(\)](#)
- **Access** public

void function table::setLinkAction(\$action) [*line* [235](#)]

Function Parameters:

- *String* \$action

defines the action type for the linking of identifier

eg. index.php?action={show}&id=12

- **Access** public

`void function table::setLinkField($field) [line 244]`

Function Parameters:

- *String* **\$field** name of the field to use

defines the field being used the will supply the action type for the linking of identifier

- **Access** public

`void function table::setPrimaryId($id) [line 272]`

Function Parameters:

- *unknown_type* **\$id**

Change primary id for the table

- **Access** public

`void function table::setRowClassFieldName($name) [line 289]`

Function Parameters:

- *String* **\$name**

Define the Column that should be define a class

- **Access** public

void function table::setRowClassName(\$name) [*line* [281](#)]

Function Parameters:

- *String* \$name

Set the Class name for a Row in the table

- **Access** public

void function table::setTableName(\$name) [*line* [253](#)]

Function Parameters:

- *\$name* \$name

Set the name of the table, need if you have more then one table on a page

- **Access** public

void function table::__destruct() [*line* [126](#)]

Destructor for this table class

- **Access** public

Class template

[[line 43](#)]

This class is used to take a input to generate templates

- **Package** PeopleScope
- **Sub-Package** Base

template::\$assigned

Array = array() [[line 54](#)]

array of assigned values to a template

template::\$repeatRegion

Array = array() [[line 67](#)]

Array of Reapeat region

- **TODO** should be removed as we are not using Dreamweaver template anymore

template::\$replacer

Array = array() [[line 60](#)]

Array of values that should be replaced in template

template::\$template_file

String = [[line 48](#)]

Location of template file

Constructor *true* function template::__construct([\$file = null]) [[line 75](#)]

Function Parameters:

- *String* **\$file** Path to current template file

Constructor for template class

- **Access** public

Constructor *true* function template::template([\$file = null]) [[line 86](#)]

Function Parameters:

- *String* **\$file** Path to current template file

Constructor php4 for template class

- **Access** public

true function template::assign(\$name, \$content) [[line 122](#)]

Function Parameters:

- *String* **\$name** name of element
- *String* **\$content** content to be replced with

Used to assign a value element to a generated template

- **Access** public

true function template::assignArray(\$assignedArray) [[line 176](#)]

Function Parameters:

- **Array \$assignedArray** array of element to be displayed

An Array of elements to be added to the template `array{'jason'=>'john', 'age'=>'14'}`

- **Access** public

true; function template::assignBlank(\$content) [[line 138](#)]

Function Parameters:

- **String \$content** element to be displayed

Assign Blank is used if not using a template, example such as XML output
example:

```
$template = new template();
$template2->assignBlank('This is a test');
```

- **Access** public

true function template::assignRepeat(\$name, \$content) [[line 164](#)]

Function Parameters:

- **String \$name** Repeat assignment name

- **Array \$content** Array of arrays values

Will assign to a repeating element from an assigned array based on array row and element name eg.

```

1      array{
2          [0]=> array{
3              [name]=>  'john'
4              [age]=>   '14'
5          }
6          [1]=> array{
7              [name]=>  'simon'
8              [age]=>   '23'
9          }
10     }
```

- **Access public**

String function template::BuildAssigned(\$content) [[line 202](#)]

Function Parameters:

- **\$content \$content** Content of the template

Build template with assigned values

- **Access private**

void function template::display() [[line 307](#)]

Will print out the full generated page template

- **Access public**

String function `template::fetch()` [[line 275](#)]

Will return the full generated page template as a variable

- **Access** public

true function `template::replace($string, [$value = "])` [[line 190](#)]

Function Parameters:

- *String* **\$string** Sting to find
- *String* **\$value** Value to be replaced with

Will replace the string across the entire template

replace happens before compiling, so it is possible to change a tag on the fly

- **Access** public

true function `template::set($file)` [[line 99](#)]

Function Parameters:

- *String* **\$file** new path to template

Set the current template path to a new template file

- **Access** public

Appendix A - Class Trees

Package PeopleScope

CustomException

- Exception
 - [CustomException](#)

db

- [db](#)

email

- [email](#)

table

- [table](#)
 - [form](#)

template

- [template](#)

Appendix D - Todo List

In Package PeopleScope

In [form::\\$autoRefresh](#)

- should be removed for JQuery ready({})

In [form::\\$autoRefreshcheckboxs](#)

- should be removed for JQuery ready({})

In [form::\\$autoRefreshInputEnter](#)

- should be removed for JQuery ready({})

In [template::\\$repeatRegion](#)

- should be removed as we are not using Dreamweaver template anymore

In [convertUldate\(\)](#)

- remember how this works :S

In [form::\\$formScript\(\)](#)

- should be removed for JQuery ready({})

In [table::getFilterType\(\)](#)

- Add <> to VALUE list

In [form::setAutoRefresh\(\)](#)

- should be removed for JQuery ready({})

In [email::setTXTtemplate\(\)](#)

- confirm this is correct

In [form::unsetAutoRefresh\(\)](#)

- should be removed for JQuery ready({})

Index

C

constructor form::__construct()	0
<i>Constructor</i>	
constructor email::email()	0
<i>Constructor for Email php4</i>	
constructor form::form()	0
<i>Constructor php4</i>	
constructor table::__construct()	0
<i>Constructor for this table class</i>	
constructor template::template()	0
<i>Constructor php4 for template class</i>	
constructor template::__construct()	0
<i>Constructor for template class</i>	
constructor db::__construct()	0
<i>constructor to initiate database conection</i>	
CustomException::queryError()	0
<i>Will append the query being used to the begining of a logError output</i>	
convertUIdate()	0
<i>Will convert the output from a JQuery UI date field format</i>	
convertMinutes2Hours()	0
<i>Will take in a amount in minutes and return the value in the amount of hours : minute</i>	
createDateField()	0
<i>This will take in a name for the field types and a date that you wish and format an option list for select boxes, and return them in an associated array for year, month and day</i>	
CustomException	0
<i>Generates a custom exception</i>	
CustomException::logError()	0
<i>Generate a formatted excption error</i>	
convertDecimalToMinutes()	0
<i>Will take in a percent amount and return values in mins</i>	

D

db::prepareToQuery()	0
<i>Merge query template with array</i>	
db::insert()	0
<i>Will instigate a INSERT query and return the inserts autocomplete Id;</i>	
db::query()	0
<i>Will instigate a query and return a recordset;</i>	
db::select()	0
<i>Will instigate a SELECT query and return and an array of responses</i>	
db::update()	0
<i>Will instigate a UPDATE query and return true if no problems;</i>	
db::getDNSString()	0

<i>returns current DSN string used to connect ot the server</i>	0
db::delete()	0
<i>Will instigate a DELETE query and return true if no problems;</i>	
db	0
<i>This class is Database abstraction layer</i>	
databaseToUI()	0
<i>converts a database date value and conversts to Au date format</i>	
db::\$dbh	0
db::\$dsn	0
db::\$lastQuery	0
database.class.php	0
<i>Database Class,</i>	
<i>
</i>	
<i>This class is Database abstraction layer
</i>	

E

email::sender()	0
<i>Set the Sender info for header</i>	
email::send()	0
<i>compile and Send email</i>	
email::recordEmail()	0
<i>Record email into the database</i>	
email::setHeader()	0
<i>Set all other Header information as required</i>	
email::setHTMLtemplate()	0
<i>Create a HTML email using the base template class</i>	
email::construct()	0
<i>Constructor for Email php5</i>	
email::subject()	0
<i>Set the Subject line fo the email</i>	
email::setTXTtemplate()	0
<i>Will create and output from Form Class standardised format String</i>	
<i>To create a TEXT email in a reable format</i>	
email::append_text()	0
<i>converts the TEXT version of the email to be sent to be appended to the email</i>	
email::append_html()	0
<i>converts the HTML version of the email to be sent to be appended to the email</i>	
email::\$CrLf	0
<i>Carrage return</i>	
email	0
<i>This class is used create and send email
</i>	
errorhandler.class.php	0
<i>CustomException Class,</i>	
<i>
</i>	
<i>Generates a custom exception
</i>	
email::\$header	0
<i>Mail Header information</i>	
email::\$mail	0
<i>Mail Object</i>	
email::append_file()	0
<i>converts the TEXT version of the email to be sent to be appended to the email</i>	

email::\$template	0
<i>Template Object</i>	
email::\$mime	0
<i>Set Mime Type</i>	
email.class.php	0
<i>Email Class,</i>	
<i>This class is used create and send email, can also record emails in a db, and can use the template class to format Html
</i>	

F

form::formFooter()	0
<i>Setup closing form tag</i>	
form::formHeader()	0
<i>Setup the form tag ready for the inputs</i>	
form::draw()	0
<i>Will Generated the required input fields from the \$this->form information information can be broken up using the string '-----'</i>	
form::\$formScript	0
<i>Java scripts reloaded to this class</i>	
form::\$form	0
<i>Holds string value for form builder</i>	
form::formScript()	0
<i>Set onSubmit Form scripts</i>	
form::freeFormSubmit()	0
<i>Standards Submit Button</i>	
form::unsetAutoRefresh()	0
<i>Remove the AutoRefresh function command</i>	
form::SetFckConfigUrl()	0
<i>Set a alternate FCK config file path</i>	
form::setAutoRefresh()	0
<i>Set the AutoRefresh function command</i>	
form::inputfield()	0
<i>Takes in a standard single row from \$this->form information and generated and html form input string</i>	
form::\$fck_width	0
<i>Width size of fck editor</i>	
form::\$fck_height	0
<i>Height size of fck editor</i>	
formatDateUI()	0
<i>Will take a Au date from jQuery Datepicker and convert to a database date format</i>	
form	0
<i>This class is used to take a input String in a standardised format and convert to a Html Form</i>	
formatDateResponce()	0
<i>This will take in an Array or if no array give will default to the \$_REQUEST</i>	
formatArray()	0
<i>Will input an associate array and output in a readable format</i>	
fileExt()	0
<i>return file extention based on Mime type for common doc types</i>	
<i>doc, docx, pdf</i>	
form::\$autoRefresh	0

JavaScript handle onChange for Form input types command	0
form::\$autoRefreshcheckboxs	0
JavaScript handle onClick for Form input types command	0
form::\$fck_configUrl	0
Url Tor find fck config file	0
form::\$enctype	0
Form enctype type	0
form::\$db	0
Database object	0
form::\$autoRefreshInputEnter	0
JavaScript handle onkeyup for Form input types command	0
form.class.php	0
Form Class,	

	
This class is used to take a input String in a standardised format and convert to a Html Form	

	

G

global \$GLOBALS['trace']	0
Global trace debugging	
global \$GLOBALS['style']	0
Global style for error messages	

P

pp()	0
A debugging tool	
parseArray()	0
Will take in a String and a start and end delimiter and will return the content between the delimiter	

Q

question.class.php	0
------------------------------------	---

S

stripElements()	0
Will take in a list of accociated array names and remove from that array via reference	
showVars()	0
Will return a html form with all the results of the \$GLOBAL vars	
showTrace()	0
Output \$GLOBAL['trace'] Trace	

T

table::setRowClassName()	0
Set the Class name for a Row in the table	
table::setRowClassFieldName()	0
Define the Column that should be define a class	
table::setTableName()	0
Set the name of the table, need if you have more then one table on a page	
table::destruct()	0
Destructor for this table class	
template	0
This class is used to take a input to generate templates	
table::setPrimaryId()	0
Change primary id for the table	
table::setLinkField()	0
defines the field being used the will supply the action type for the linking of identifier	
table::setHeader()	0
Will using an input of an associated array to rewrite field names to more readable header, if no match is found the header will be the field name	
table::setFooter()	0
Set the table Footer row for the table Html Footer rules apply	
table::setIdIdentifier()	0
The name of the field that should be used to identify the row	
table::setIdIdentifierPage()	0
A page that the setIdIdentifier should goto if selected	
table::setLinkAction()	0
defines the action type for the linking of identifier	
template::\$assigned	0
array of assigned values to a template	
template::\$repeatRegion	0
Array of Repeat region	
template::display()	0
Will print out the full generated page template	
template::BuildAssigned()	0
Build template with assigned values	
template::fetch()	0
Will return the full generated page template as a variable	
template::replace()	0
Will replace the string across the entire template	
template::set()	0
Set the current template path to a new template file	
template::assignRepeat()	0
Will assign to a repeating element from an assigned array based on array row and element name eg.	
template::assignBlank()	0
Assign Blank is used if not using a template, example such as XML output	
template::\$replacer	0
Array of values that should be replaced in template	
template::\$template_file	0
Location of template file	
template::assign()	0
Used to assign a value element to a generated template	
template::assignArray()	0
An Array of elements to be added to the template	

<code>array(['jason'=>'john', 'age'=>'14']</code>	0
<code>table::setFilter()</code>	0
<i>Will using an input of an associated array add apply a filter type to a field name</i>	
<code>table::setColumnsWidth()</code>	0
<i>Associated array of Name of field for keys and the value is width of the field HTML rules apply, width in pixels '24', width percent '50%', width flexible '*'</i>	
<code>table::\$headerArray</code>	0
<i>An Associated array of Key: fields names, Value: new column name</i>	
<code>table::\$footer</code>	0
<i>Define a footer row for your table</i>	
<code>table::\$id</code>	0
<i>String \$id name of table</i>	
<code>table::\$identifier</code>	0
<i>name of field that define the unique identifier for the table</i>	
<code>table::\$link_action</code>	0
<i>Define &action type for page if identifier is set</i>	
<code>table::\$filterArray</code>	0
<i>An Associated array of Key: fields names, Value: Filter type</i>	
<code>table::\$columnsWidth</code>	0
<i>An Associated array of Key: fields names, Value: column with</i>	
<code>tools.function.php</code>	0
<i>Tools Function,
 At set of general tool to help with development
</i>	
<code>template.class.php</code>	0
<i>Template Class,
 This class is used to take a input to generate templates
</i>	
<code>trace()</code>	0
<i>Debugging tool that that will store the outcomes into a \$GLOBAL['trace'] var</i>	
<code>table</code>	0
<i>This class is used to take the output from a select Query and turn it into a filterable table
</i>	
<code>table::\$basePage</code>	0
<i>name of the file that is link to from identifier</i>	
<code>table::\$link_field</code>	0
<i>Define &action based on a fields name for page if identifier is set</i>	
<code>table::\$name</code>	0
<i>Name of table, used for id field</i>	
<code>table::getFilterType()</code>	0
<i>This function will create a filter for a field based on its type</i>	
<code>table::generateDisplayTable()</code>	0
<i>This function will take the output of database query and generate a table for it, including headers and filters</i>	
<code>table::getOrderType()</code>	0
<i>Will wrap div information around the header name so we can add id for Asc Desc triggers for ordering</i>	
<code>table::removeColumn()</code>	0
<i>An array of fields that should not be shown in the table</i>	
<code>table::setBasePage()</code>	0
<i>if using identifier then define the page the it links to</i>	

<i>Default is the current page</i>	
table::buildWhereArrayFromRequest()	0
<i>This function will take in the information from the \$_REQUEST global vars from a filter and conver it to a WHERE statement to be append to a query to filter the output</i>	
table::buildTd()	0
<i>This function will take the field values and wrap a td tag around it, It will also format the field based on the filter type</i>	
table::\$nolink	0
<i>define if table links to another page for each row</i>	
table::\$removeColumnArray	0
<i>A list of field names in an arrays to be removed from the table columns</i>	
table::\$row_class field name	0
<i>Define a class for a column</i>	
table::\$row_class_name	0
<i>defined class name for the all the rows</i>	
table.class.php	0
<i>Table Class,
 This class is used to take the output from a select Query and turn it into a filterable table
</i>	