People_scope API Manual

Contents

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 referance

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 formatDateResponce($name, [&$listArray = NULL]) *[line 241]*
Function Parameters:

- *String* **$name** the name ofthe fields we are looking for
- *Array* **&$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 accociated 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 todays date accociated 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 debuging 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* **$elements** Comma seperated list of names
- *Array* **$array** The array to remove them from via referance
- **&$array**

Will take in a list of accociated array names and remove from that array via referance

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

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

- **See** showTrace for output

$GLOBALS['style']

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

Global style for error messages

$GLOBALS['trace']

string = array() *[line 24]*

Global trace debugging

Class CustomException

[line 34]

Generates a custom exception

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

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

Generate a formated 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 instegate 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 and 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** identifer 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 varable converstion

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 identifer

- **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 roe fro 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 unquie identfier for the table

- **Var** Id feild
- **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::genterateDisplayTable($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 identifer then define the page the it links to Default is the current page

- **Access** public

void function table::setColumnsWidth($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 flexable '*'

- **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 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

- **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 rewite field names to more readable header, if no match is found the header will be the field name
An Accociated 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 isself
with a referance 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 setIdentifer 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 identifer
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 identifer

- **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](281)]*
Function Parameters:

- *String* **$name**

Set the Class name for a Row in the table

- **Access** public

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

Contructor for template class

- **Access** public

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

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

Contructor 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 bassed on array row and element name eg.

```
array{
        [0]=> array{
                [name]=>   'john'
                [age]=>   '14'
        }
        [1]=> array{
                [name]=>   'simon'
                [age]=>   '23'
        }
}
```

- **Access** public

Sting 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

Sting function template::fetch() *[line 275]*

Will return the full generated page template as a variarable

- **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 **convertUIdate()**:

- 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