

# People Scope API Document



# Contents

<a href="#">Package default Procedural Elements</a>	2
<a href="#">template.class.php</a>	2
<a href="#">login.class.php</a>	3
<a href="#">Package default Classes</a>	4
<a href="#">Class login</a>	4
<a href="#">Var \$admin</a>	4
<a href="#">Var \$error</a>	4
<a href="#">Var \$lastError</a>	4
<a href="#">Var \$table</a>	5
<a href="#">Var \$template</a>	5
<a href="#">Constructor construct</a>	5
<a href="#">Method checkUserLogin</a>	5
<a href="#">Method getHomePage</a>	5
<a href="#">Class template</a>	5
<a href="#">Var \$db</a>	6
<a href="#">Var \$db connect</a>	6
<a href="#">Var \$filterArray</a>	6
<a href="#">Var \$headerArray</a>	6
<a href="#">Var \$layout</a>	6
<a href="#">Var \$template</a>	7
<a href="#">Constructor construct</a>	7
<a href="#">Method assign</a>	7
<a href="#">Method content</a>	7
<a href="#">Method display</a>	8
<a href="#">Method externalLink</a>	8
<a href="#">Method fetch</a>	8
<a href="#">Method formatBoolean</a>	8
<a href="#">Method formatValue</a>	9
<a href="#">Method getListTable</a>	9
<a href="#">Method input</a>	9
<a href="#">Method insert</a>	10
<a href="#">Method page</a>	10
<a href="#">Method strip_tags</a>	10
<a href="#">Method destruct</a>	11
<a href="#">Package PeopleScope Procedural Elements</a>	13
<a href="#">advertisement.class.php</a>	13
<a href="#">advertTemplate.class.php</a>	14
<a href="#">errorhandler.class.php</a>	15
<a href="#">Function exception_error_handler</a>	15
<a href="#">Function exception_handler</a>	15
<a href="#">Function myErrorHandler</a>	15
<a href="#">template.old.class.php</a>	17
<a href="#">category.class.php</a>	18

<a href="#">employmenttype.class.php</a>	19
<b><a href="#">Package PeopleScope Classes</a></b>	20
<a href="#">Class advertisement</a>	20
<a href="#">Var \$db</a>	20
<a href="#">Var \$db_connect</a>	20
<a href="#">Var \$fields</a>	21
<a href="#">Var \$fields_required</a>	21
<a href="#">Var \$fields_validation_type</a>	21
<a href="#">Var \$table</a>	22
<a href="#">Var \$template</a>	22
<a href="#">Var \$validation_error</a>	22
<a href="#">Constructor construct</a>	22
<a href="#">Method create</a>	23
<a href="#">Method createAdvertisementDetails</a>	24
<a href="#">Method deleteAdvertisementDetails</a>	24
<a href="#">Method editAdvertisementDetails</a>	25
<a href="#">Method getAdvertisementList</a>	25
<a href="#">Method getSelectListOfCategory</a>	26
<a href="#">Method getSelectListOfEmploymentType</a>	26
<a href="#">Method getSelectListOfStates</a>	27
<a href="#">Method getSelectListOfTemplate</a>	27
<a href="#">Method getUserById</a>	28
<a href="#">Method lists</a>	28
<a href="#">Method read</a>	29
<a href="#">Method remove</a>	29
<a href="#">Method saveAdvertisementDetails</a>	30
<a href="#">Method showAdvertisementDetails</a>	30
<a href="#">Method templateAdvertisementLayout</a>	31
<a href="#">Method update</a>	31
<a href="#">Method updateAdvertisementDetails</a>	32
<a href="#">Method Validate</a>	33
<a href="#">Class advertTemplate</a>	33
<a href="#">Var \$db</a>	34
<a href="#">Var \$db_connect</a>	34
<a href="#">Var \$fields</a>	34
<a href="#">Var \$fields_required</a>	35
<a href="#">Var \$fields_validation_type</a>	35
<a href="#">Var \$table</a>	35
<a href="#">Var \$template</a>	36
<a href="#">Var \$validation_error</a>	36
<a href="#">Constructor construct</a>	36
<a href="#">Method create</a>	36
<a href="#">Method createAdvertTemplateDetails</a>	37
<a href="#">Method deleteAdvertTemplateDetails</a>	38
<a href="#">Method editAdvertTemplateDetails</a>	38
<a href="#">Method getAdvertTemplateList</a>	39
<a href="#">Method getSelectListOfCategory</a>	39
<a href="#">Method getSelectListOfEmploymentType</a>	40
<a href="#">Method getSelectListOfStates</a>	40

<a href="#">Method lists</a>	41
<a href="#">Method read</a>	41
<a href="#">Method remove</a>	42
<a href="#">Method saveAdvertTemplateDetails</a>	42
<a href="#">Method showAdvertTemplateDetails</a>	43
<a href="#">Method templateAdvertTemplateLayout</a>	43
<a href="#">Method update</a>	44
<a href="#">Method updateAdvertTemplateDetails</a>	45
<a href="#">Method Validate</a>	45
<a href="#">Class category</a>	46
<a href="#">Var \$db</a>	46
<a href="#">Var \$db_connect</a>	47
<a href="#">Var \$fields</a>	47
<a href="#">Var \$fields_required</a>	47
<a href="#">Var \$fields_validation_type</a>	48
<a href="#">Var \$table</a>	48
<a href="#">Var \$template</a>	48
<a href="#">Var \$validation_error</a>	48
<a href="#">Constructor construct</a>	49
<a href="#">Method create</a>	49
<a href="#">Method createCategoryDetails</a>	50
<a href="#">Method deleteCategoryDetails</a>	50
<a href="#">Method editCategoryDetails</a>	51
<a href="#">Method getCategoryList</a>	51
<a href="#">Method lists</a>	52
<a href="#">Method read</a>	53
<a href="#">Method remove</a>	53
<a href="#">Method saveCategoryDetails</a>	53
<a href="#">Method showCategoryDetails</a>	54
<a href="#">Method templateCategoryLayout</a>	55
<a href="#">Method update</a>	55
<a href="#">Method updateCategoryDetails</a>	56
<a href="#">Method Validate</a>	57
<a href="#">Class employmenttype</a>	57
<a href="#">Var \$db</a>	58
<a href="#">Var \$db_connect</a>	58
<a href="#">Var \$fields</a>	58
<a href="#">Var \$fields_required</a>	59
<a href="#">Var \$fields_validation_type</a>	59
<a href="#">Var \$table</a>	59
<a href="#">Var \$template</a>	59
<a href="#">Var \$validation_error</a>	60
<a href="#">Constructor construct</a>	60
<a href="#">Method create</a>	60
<a href="#">Method createEmploymenttypeDetails</a>	61
<a href="#">Method deleteEmploymenttypeDetails</a>	61
<a href="#">Method editEmploymenttypeDetails</a>	62
<a href="#">Method getEmploymenttypeList</a>	62
<a href="#">Method lists</a>	63

<a href="#">Method read</a>	64
<a href="#">Method remove</a>	64
<a href="#">Method saveEmploymenttypeDetails</a>	65
<a href="#">Method showEmploymenttypeDetails</a>	65
<a href="#">Method templateEmploymenttypeLayout</a>	66
<a href="#">Method update</a>	66
<a href="#">Method updateEmploymenttypeDetails</a>	67
<a href="#">Method Validate</a>	68
<a href="#">database.class.php</a>	70
<a href="#">email.class.php</a>	71
<a href="#">form.class.php</a>	72
<a href="#">table.class.php</a>	73
<a href="#">tools.function.php</a>	74
<a href="#">Function Box</a>	74
<a href="#">Function convertDecimalToMinutes</a>	74
<a href="#">Function convertMinutes2Hours</a>	74
<a href="#">Function convertUIdate</a>	75
<a href="#">Function createDateField</a>	75
<a href="#">Function databaseToUI</a>	75
<a href="#">Function fileExt</a>	76
<a href="#">Function formatArray</a>	76
<a href="#">Function formatDateResponce</a>	76
<a href="#">Function formatDateUI</a>	76
<a href="#">Function parseArray</a>	77
<a href="#">Function pp</a>	77
<a href="#">Function showTrace</a>	77
<a href="#">Function showVars</a>	77
<a href="#">Function stripElements</a>	78
<a href="#">Function trace</a>	78
<a href="#">Function warning</a>	78
<a href="#">Global Variable \$GLOBALS['style']</a>	78
<a href="#">Global Variable \$GLOBALS['trace']</a>	78
<a href="#">Class CustomException</a>	79
<a href="#">Method logError</a>	79
<a href="#">Method messageError</a>	79
<a href="#">Method queryError</a>	79
<a href="#">Class db</a>	80
<a href="#">Var \$dbh</a>	80
<a href="#">Var \$dsn</a>	80
<a href="#">Var \$lastQuery</a>	80
<a href="#">Constructor __construct</a>	80
<a href="#">Method delete</a>	80
<a href="#">Method getDNSString</a>	81
<a href="#">Method insert</a>	81
<a href="#">Method prepareToQuery</a>	81
<a href="#">Method query</a>	82
<a href="#">Method select</a>	82
<a href="#">Method update</a>	82
<a href="#">Class email</a>	83

<a href="#">Var \$CrLf</a>	83
<a href="#">Var \$Header</a>	83
<a href="#">Var \$Mail</a>	83
<a href="#">Var \$Mime</a>	84
<a href="#">Var \$Template</a>	84
<a href="#">Constructor email</a>	84
<a href="#">Method append_file</a>	84
<a href="#">Method append_html</a>	84
<a href="#">Method append_text</a>	84
<a href="#">Method recordEmail</a>	84
<a href="#">Method send</a>	85
<a href="#">Method sender</a>	85
<a href="#">Method setHeader</a>	85
<a href="#">Method setHTMLtemplate</a>	85
<a href="#">Method setTXTtemplate</a>	86
<a href="#">Method subject</a>	86
<a href="#">Method construct</a>	86
<a href="#">Class form</a>	87
<a href="#">Var \$autoRefresh</a>	87
<a href="#">Var \$autoRefreshcheckboxs</a>	87
<a href="#">Var \$autoRefreshInputEnter</a>	87
<a href="#">Var \$db</a>	88
<a href="#">Var \$enctype</a>	88
<a href="#">Var \$fck_configUrl</a>	88
<a href="#">Var \$fck_height</a>	88
<a href="#">Var \$fck_width</a>	88
<a href="#">Var \$form</a>	88
<a href="#">Var \$formScript</a>	89
<a href="#">Constructor construct</a>	89
<a href="#">Constructor form</a>	89
<a href="#">Method draw</a>	89
<a href="#">Method formFooter</a>	89
<a href="#">Method formHeader</a>	89
<a href="#">Method formScript</a>	90
<a href="#">Method freeFormSubmit</a>	90
<a href="#">Method inputfield</a>	90
<a href="#">Method setAutoRefresh</a>	90
<a href="#">Method SetFckConfigUrl</a>	91
<a href="#">Method unsetAutoRefresh</a>	91
<a href="#">Class table</a>	91
<a href="#">Var \$basePage</a>	92
<a href="#">Var \$columnsWidth</a>	92
<a href="#">Var \$filterArray</a>	92
<a href="#">Var \$footer</a>	93
<a href="#">Var \$headerArray</a>	93
<a href="#">Var \$id</a>	93
<a href="#">Var \$identifier</a>	93
<a href="#">Var \$link_action</a>	94
<a href="#">Var \$link_field</a>	94

<a href="#">Var \$name</a>	94
<a href="#">Var \$nolink</a>	94
<a href="#">Var \$removeColumnArray</a>	94
<a href="#">Var \$rowsOnly</a>	95
<a href="#">Var \$row class field name</a>	95
<a href="#">Var \$row class name</a>	95
<a href="#">Var \$SEUrl</a>	95
<a href="#">Constructor construct</a>	95
<a href="#">Method buildTd</a>	95
<a href="#">Method buildWhereArrayFromRequest</a>	96
<a href="#">Method generateDisplayTable</a>	96
<a href="#">Method getFilterType</a>	96
<a href="#">Method getOrderType</a>	97
<a href="#">Method removeColumn</a>	97
<a href="#">Method setBasePage</a>	97
<a href="#">Method setColumnsClass</a>	98
<a href="#">Method setColumnsWidth</a>	98
<a href="#">Method setFilter</a>	98
<a href="#">Method setFooter</a>	98
<a href="#">Method setHeader</a>	99
<a href="#">Method setIdentifier</a>	99
<a href="#">Method setIdentifierPage</a>	99
<a href="#">Method setLinkAction</a>	99
<a href="#">Method setLinkField</a>	100
<a href="#">Method setPrimaryId</a>	100
<a href="#">Method setRowClassFieldName</a>	100
<a href="#">Method setRowClassName</a>	101
<a href="#">Method setRowsOnly</a>	101
<a href="#">Method setTableName</a>	101
<a href="#">Method destruct</a>	101
<a href="#">Class template</a>	102
<a href="#">Var \$assigned</a>	102
<a href="#">Var \$repeatRegion</a>	102
<a href="#">Var \$replacer</a>	102
<a href="#">Var \$template file</a>	102
<a href="#">Constructor construct</a>	102
<a href="#">Constructor template</a>	103
<a href="#">Method assign</a>	103
<a href="#">Method assignArray</a>	103
<a href="#">Method assignBlank</a>	104
<a href="#">Method assignRepeat</a>	104
<a href="#">Method BuildAssigned</a>	105
<a href="#">Method display</a>	105
<a href="#">Method fetch</a>	105
<a href="#">Method replace</a>	106
<a href="#">Method set</a>	106
<a href="#">Appendices</a>	107
<a href="#">Appendix A - Class Trees</a>	108
<a href="#">PeopleScope</a>	108

<a href="#">default</a>	109
<a href="#">Appendix C - Source Code</a>	110
<a href="#">Package PeopleScope</a>	111
<a href="#">source code: template.class.php</a>	112
<a href="#">source code: login.class.php</a>	115
<a href="#">Package PeopleScope</a>	116
<a href="#">source code: advertisement.class.php</a>	117
<a href="#">source code: advertTemplate.class.php</a>	132
<a href="#">source code: errorHandler.class.php</a>	146
<a href="#">source code: template.old.class.php</a>	149
<a href="#">source code: category.class.php</a>	154
<a href="#">source code: employmenttype.class.php</a>	165
<a href="#">source code: database.class.php</a>	176
<a href="#">source code: email.class.php</a>	180
<a href="#">source code: form.class.php</a>	184
<a href="#">source code: table.class.php</a>	189
<a href="#">source code: tools.function.php</a>	195
<a href="#">Appendix D - Todo List</a>	201





# Package default Procedural Elements

template.class.php

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

# login.class.php

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

# Package default Classes

## Class login

[[line 3](#)]

- **Package** default

**login::\$admin**

*mixed* = [[line 9](#)]

- **Access** public

**login::\$error**

*mixed* = [[line 8](#)]

- **Access** private

**login::\$lastError**

*mixed* = [[line 5](#)]

- **Access** public

#### login::\$table

*mixed* = [[line 6](#)]

- **Access** public

#### login::\$template

*mixed* = [[line 7](#)]

- **Access** public

Constructor *void* function login::\_\_construct() [[line 11](#)]

*void* function login::checkUserLogin(\$uname, \$password) [[line 24](#)]

#### **Function Parameters:**

- **\$uname**
- **\$password**

*void* function login::getHomePage() [[line 17](#)]

- **Access** public

## Class template

[[line 4](#)]

- **Package** default

#### **template::\$db**

*mixed* = [line [7](#)]

- **Access** private

#### **template::\$db\_connect**

*mixed* = [line [6](#)]

- **Access** private

#### **template::\$filterArray**

*mixed* = [line [10](#)]

- **Access** private

#### **template::\$headerArray**

*mixed* = [line [9](#)]

- **Access** private

#### **template::\$layout**

*mixed* = 'layout.tpl.html' [line [8](#)]

- **Access** private

## **template::\$template**

*mixed* = [line [11](#)]

- **Access** private

Constructor *void* function template::\_\_construct([\$layout = NULL]) [line [15](#)]

### **Function Parameters:**

- **\$layout**

- **Access** public

*void* function template::assign(\$field, \$value, [&\$tpl = NULL]) [line [60](#)]

### **Function Parameters:**

- **\$field**
- **\$value**
- **&\$tpl**

- **Access** public

*void* function template::content(\$field) [line [55](#)]

### **Function Parameters:**

- **\$field**

- **Access** public

*void* function `template::display()` [*line* [78](#)]

- **Access** public

*void* function `template::externalLink($link)` [*line* [158](#)]

**Function Parameters:**

- **\$link**

- **Access** public

*void* function `template::fetch([&$tpl = NULL])` [*line* [69](#)]

**Function Parameters:**

- **&\$tpl**

- **Access** public

*void* function `template::formatBoolean($value)` [*line* [82](#)]

**Function Parameters:**



- **\$value**

- **Access** public

*void* function `template::formatValue($value, $msg)` [*line* [92](#)]

**Function Parameters:**

- **\$value**
- **\$msg**

- **Access** public

*void* function `template::getListTable($table, $value, $idField, $valueField, [$selectBox = NULL], [$WHERE = NULL])` [*line* [102](#)]

**Function Parameters:**

- **\$table**
- **\$value**
- **\$idField**
- **\$valueField**
- **\$selectBox**
- **\$WHERE**

- **Access** public

*void* function `template::input($type, $name, [$value = NULL])` [*line* [138](#)]

**Function Parameters:**

- **\$type**
- **\$name**
- **\$value**

- **Access** public

*void* function `template::insert($template)` [*line* [50](#)]

***Function Parameters:***

- **\$template**

- **Access** public

*void* function `template::page($field)` [*line* [44](#)]

***Function Parameters:***

- **\$field**

- **Access** public

*void* function `template::strip_tags($string)` [*line* [152](#)]

***Function Parameters:***

- **\$string**

- **Access** private

*void* function template::\_\_destruct() [*line* [40](#)]

- **Access** public



# Package PeopleScope Procedural Elements

## advertisement.class.php

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

# advertTemplate.class.php

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

## 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('SELECT * FROM notable ');  
}catch(CustomException $e){  
    echo $e->logError();  
}
```

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

*void function exception\_error\_handler(\$errno, \$errstr, \$errfile, \$errline) [line [85](#)]*

**Function Parameters:**

- **\$errno**
- **\$errstr**
- **\$errfile**
- **\$errline**

*void function exception\_handler(\$exception) [line [91](#)]*

**Function Parameters:**

- **\$exception**

*void function myErrorHandler(\$errno, \$errstr, \$errfile, \$errline) [line [97](#)]*

***Function Parameters:***

- **\$errno**
- **\$errstr**
- **\$errfile**
- **\$errline**



## template.old.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](mailto:jason@lexxcom.com.au)>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

require\_once ['errorhandler.class.php'](#) *line 35*

### required error handler

## category.class.php

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

# employmenttype.class.php

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

# Package PeopleScope Classes

## Class advertisement

[line [15](#)]

### Advertisement Class

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div> </pre>

- **Package** PeopleScope
- **Author** Jennifer Erator < [jason@lexxcom.com](mailto:jason@lexxcom.com)>
- **Version** 1.1 of the Framework generator

### advertisement::\$db

Object = [line [27](#)]

### Database class object

- **Access** private

### advertisement::\$db\_connect

Object = [line [21](#)]

### Connect to PDO object through database class

- **Access** private

#### **advertisement::\$fields**

```
Array = array('advertisement_id', 'title', 'catagory_id', 'template_id', 'office_id', 'dept_id', 'role_id', 'state_id',
'store_location_id', 'storerole_id', 'start_date', 'end_date', 'discription', 'requirments', 'upload_resume',
'cover_letter', 'status', 'employmenttype_id', 'create_date', 'create_by', 'modify_date', 'modify_by',
'delete_date', 'delete_by') [line 45]
```

**Array of field used in the database if not in this list is dropped from insert or update**

- **Access** private

#### **advertisement::\$fields\_required**

```
Array|null = array('title') [line 51]
```

**Array of feilds require information when validating**

- **Access** private

#### **advertisement::\$fields\_validation\_type**

```
Array|null = array ('advertisement_id'=>'TEXT', 'title'=>'TEXT', 'catagory_id'=>'INT', 'template_id'=>'INT',
'office_id'=>'INT', 'dept_id'=>'INT', 'role_id'=>'INT', 'state_id'=>'INT', 'store_location_id'=>'INT',
'storerole_id'=>'INT', 'start_date'=>'DATE', 'end_date'=>'DATE', 'discription'=>'TEXT',
'requirments'=>'TEXT', 'upload_resume'=>'BOOL', 'cover_letter'=>'BOOL', 'status'=>'BOOL',
'employmenttype_id'=>'INT', 'create_date'=>'TEXT', 'create_by'=>'INT', 'modify_date'=>'TEXT',
'modify_by'=>'INT', 'delete_date'=>'TEXT', 'delete_by'=>'TEXT') [line 57]
```

**Array of feilds and there types that are check when validating**

- **Access** private

#### **advertisement::\$table**

*Object = [line [33](#)]*

#### **Table class object**

- **Access** public

#### **advertisement::\$template**

*Object = [line [39](#)]*

#### **Template class object**

- **Access** public

#### **advertisement::\$validation\_error**

*Array = array() [line [64](#)]*

#### **Array use to store any error found during Validation function**

- **See** Validation()
- **Access** private

Constructor *void* function advertisement::\_\_construct() [line [82](#)]

#### **Constructor for this method**

The constructor will setup the required object for this class will initiate the database class, the table class and the template

for this class to use

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [advertisement::\\$template](#)
- See [advertisement::\\$table](#)
- See db::
- Access public

*Integer function advertisement::create(\$source) [line [204](#)]*

**Function Parameters:**

- Array **\$source**

**This method will take an array and insert it in the database**

This method will insert the formatted information into a database, the format for the array should be an associated array being the first key should be the table inserting with the keys for child array the fields that are being inserted too and the values to insert

```
Array
(
    [users] => Array
        (
            [name] => Dave
            [surname] => Smith
            [email] => dave@dave.com
        )
    [staff] => Array
        (
            [staff_id] => 1245
            [office_number] => 22
            [drown_code] => bee223
        )
)
```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*void* function advertisement::createAdvertisementDetails() [*line* [622](#)]

**This method will provide a page to the to add a single row Advertisement to the advertisement table**

The method using the template class to format the information ready to display the the user, so that they may be able to add any of the fields releated to a row in the database. The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function advertisement::deleteAdvertisementDetails(\$id) [*line* [723](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to marked as delete

**Set a row to be marked as deleted**

This method will take the id and set the delete\_date field to the current datetime, which will marking it as deleted

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public



*void function advertisement::editAdvertisementDetails(\$id) [line 520]*

**Function Parameters:**

- **Integer \$id** The primary id of the row to show

**Show the details ready to edit of a single Advertisement from the advertisement**

This method is used to display and editable page to the use, so that they maybe able to edit any of the fields releated to the row in question.

The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void function advertisement::getAdvertisementList([\$type = 'TABLE'], [\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [line 417]*

**Function Parameters:**

- **String \$type** Option of type of response for the output of the list
- **String \$orderby** Which single field is used to oder the output
- **String \$direction** Which direction os required for the orderby output
- **Array \$filter** A array of fields to filter, key=INT set (eg array('tile'=>'this title'))

**Show list of information corresponding the to this class**

This Method will produce a list of all the element corresponding to the result of Advertisement

using the base/table.class.php file, which will format the list into a filtable table that uses ajax class to change the content on filtering

There are to response type for this table for the parameter \$type

TABLE = Will return the content in a table with a filter row and a heading row

AJAX = Will return just the content after evaluating the filter or heading infomation

The parameter \$filter expects an array with the key being the field to look for and the value being the the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*string* function advertisement::getSelectListOfCategory(\$id, [\$selectBox = NULL]) [*line* [889](#)]

**Function Parameters:**

- *Integer* **\$id** the id of the row we are looking for
- *Boolean* **\$selectBox** True will return it as a select box with field selected, else just the field

**This Method will either the catagory\_name feild or a select box of catagory\_name fields**

This Methos is based in the catagory\_id of the category table, it will return either the information in the catagory\_name field as a string or a select box with the id selected

- **TODO** move this to its own class for all classes to use
- **Access** public

*string* function advertisement::getSelectListOfEmploymentType(\$id, [\$selectBox = NULL]) [*line* [939](#)]

**Function Parameters:**

- *Integer* **\$id** the id of the row we are looking for
- *Boolean* **\$selectBox** True will return it as a select box with field selected, else just the field

**This Method will either the employmenttype feild or a select box of employmenttype fields**

This Methos is based in the employmenttype\_id of the employmenttype table, it will reurn either the information in the employmenttype field as a string or a select box with the id selected

- **TODO** move this to its own class for all classes to use
- **Access** public

*string* function advertisement::getSelectListOfStates(\$id, [\$selectBox = NULL]) [*line* [914](#)]

**Function Parameters:**

- *Integer* **\$id** the id of the row we are looking for
- *Boolean* **\$selectBox** True will return it as a select box with field selected, else just the field

**This Method will either the name feild or a select box of state fields**

This Methos is based in the state\_id of the state table, it will reurn either the information in the state name field as a string or a select box with the id selected

- **TODO** move this to its own class for all classes to use
- **Access** public

*string* function advertisement::getSelectListOfTemplate(\$id, [\$selectBox = NULL]) [*line* [965](#)]

**Function Parameters:**

- *Integer* **\$id** the id of the row we are looking for
- *Boolean* **\$selectBox** True will return it as a select box with field selected, else just the field

**This Method will either the employmenttype feild or a select box of employmenttype fields**

This Methos is based in the employmenttype\_id of the employmenttype table, it will reurn either the information in the employmenttype field as a string or a select box with the id selected

- **TODO** move this to its own class for all classes to use

- **TODO** move this as a function in advertTemplate class
- **Access** public

String function advertisement::getUserById([\$id = false]) [*line 1002*]

**Function Parameters:**

- *Integer* \$id

### Will get name of the user based on id

This will return the name and surname concatenated of a user by their id

- **Access** public

void function advertisement::lists([\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [*line 115*]

**Function Parameters:**

- *String* \$orderby Which single field is used to order the output
- *String* \$direction Which direction is required for the orderby output
- *Array* \$filter A array of fields to filter, key=\$val set (eg array('title'=>'this title'))

### Show will pull a list from the corresponding Advertisement advertisement

This Method will produce a list of all the element corresponding to the result of Advertisement

I will only pull rows that are not considered delete

eg. the delete\_date field is not "0000-00-00 00:00:00" or set to NULL

The parameter \$filter expects an array with the key being the field to look for and the value being the the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

void function advertisement::read(\$id) [[line 249](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will return information as row**

This method is you to get a single row of information from the database based ith the primary id and return it as an array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

Boolean function advertisement::remove(\$id) [[line 374](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will update a row and make the recored as deleted**

This method will take the id and set the delete\_date field to the current datetime, which will marking it as deleted

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*void function advertisement::saveAdvertisementDetails(\$id) [line [655](#)]*

**Function Parameters:**

- *Integer \$id* The primary id of the row to updated

**save the information in a single Advertisement to the advertisement table**

This method is used to take the information from createDepartmentDetails and try to validate it through the validation method and on success will format it ready for inserted into the database through the insert method

if the validate fails then the user is show a page that mimics the createDepartmentDetails method and point out error in there input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [advertisement::update\(\)](#)
- See [advertisement::Validate\(\)](#)
- See createDepartmentDetails()
- Access public

*void function advertisement::showAdvertisementDetails(\$id) [line [492](#)]*

**Function Parameters:**

- *Integer \$id* the primary id of the row to show

**Show details of a single Advertisement from the advertisement**

This method will return a template page of the information requested the method use the template class to format the information ready to display the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function advertisement::templateAdvertisementLayout(\$fieldMember, [\$input = false], [\$inputArray = array()], \$fielddata) [*line* [742](#)]

**Function Parameters:**

- *Array* **\$fielddata** An associative array of fields that need to be assigned to the template object
- *Boolean* **\$input** If false then just assign the value if true the add the value to corresponding form element
- *Array* **\$inputArray** Not sure :S
- **\$fieldMember**

**This method assigns the associate array values to the template**

This method is used to incorporate the standards elements of the templates to a single function across all templatd methods

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **TODO** find out what \$inputArray is used for
- **Access** private

*Integer* function advertisement::update(\$source, \$id) [*line* [328](#)]

**Function Parameters:**

- *Array* **\$source**
- **\$id**

**This method will take an array and update a row in the database**

This method will update the formatted information into the database, the format for the array

should be an associated array being the first key should be the table to be updated with the keys

for child array the fields that are being updated too and the values to be updated

```
Array
(
    [users] => Array
        (
            [name] => Dave
            [surname] => Smith
            [email] => dave@dave.com
        )
    [staff] => Array
        (
            [staff_id] => 1245
            [office_number] => 22
            [drown_code] => bee223
        )
)
```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

void function advertisement::updateAdvertisementDetails(\$id) [[line 554](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to updated

### **update the information in a single Advertisement from the advertisement**

This method is used to take the information from editDepartmentDetails and try to validate it through the validation method and on success will format it ready for input into the database through the update method

if the validate fails then the user is show a page that mimics the editDepartmentDetails method and point out error in there input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>



- See [advertisement::update\(\)](#)
- See [advertisement::Validate\(\)](#)
- See [editDepartmentDetails\(\)](#)
- Access public

void function advertisement::Validate(\$request) [*line [794](#)*]

**Function Parameters:**

- Array **\$request**

**This method is used to validate inputs from form information**

This method will first check the if the fields are in the valid\_field array and strip out any that are not

Then it check that fields that require a value in them from the fields\_required have a value, if not add an error to validation\_error array

Then it will check all the values to find out if the value match the type found in the fields\_validation\_type array, if not add an error to validation\_error array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [advertisement::\\$validation\\_error](#)
- See [advertisement::\\$fields](#)
- See [advertisement::\\$fields\\_validation\\_type](#)
- See [advertisement::\\$fields\\_required](#)
- Access public

## Class advertTemplate

[*line [15](#)*]

### AdvertTemplate Class

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div> </pre>

- **Package** PeopleScope
- **Author** Jennifer Erator < [jason@lexxcom.com](mailto:jason@lexxcom.com)>
- **Version** 1.1 of the Framework generator

#### **advertTemplate::\$db**

*Object = [line [27](#)]*

#### **Database class object**

- **Access** private

#### **advertTemplate::\$db\_connect**

*Object = [line [21](#)]*

#### **Connect to PDO object through database class**

- **Access** private

#### **advertTemplate::\$fields**

*Array = array('template\_id', 'title', 'employmenttype\_id', 'catagory\_id', 'office\_id', 'dept\_id', 'role\_id', 'state\_id', 'storeLoc\_id', 'start\_date', 'end\_date', 'discription', 'requirments', 'status', 'tracking\_id', 'advertisement\_id', 'create\_date', 'modify\_date', 'delete\_date') [line [45](#)]*

**Array of field used in the database if not in this list is dropped from insert or update**

- **Access** private

#### **advertTemplate::\$fields\_required**

*Array|null* = array('title', 'catagory\_id') [[line 51](#)]

#### **Array of feilds require information when validating**

- **Access** private

#### **advertTemplate::\$fields\_validation\_type**

*Array|null* = array ('template\_id'=>'TEXT', 'title'=>'TEXT', 'employmenttype\_id'=>'INT', 'catagory\_id'=>'INT', 'office\_id'=>'INT', 'dept\_id'=>'INT', 'role\_id'=>'INT', 'state\_id'=>'INT', 'storeLoc\_id'=>'INT', 'start\_date'=>'DATE', 'end\_date'=>'DATE', 'discription'=>'TEXT', 'requirments'=>'TEXT', 'status'=>'TEXT', 'tracking\_id'=>'INT', 'advertisement\_id'=>'INT', 'create\_date'=>'TEXT', 'modify\_date'=>'TEXT', 'delete\_date'=>'TEXT') [[line 57](#)]

#### **Array of feilds and there types that are check when validating**

- **Access** private

#### **advertTemplate::\$table**

*Object* = [[line 33](#)]

#### **Table class object**

- **Access** public

#### **advertTemplate::\$template**

*Object = [line 39]*

## Template class object

- **Access** public

**advertTemplate::\$validation\_error**

*Array = array() [line 64]*

## Array use to store any error found during Validation function

- **See** Validation()
- **Access** private

Constructor *void* function **advertTemplate::\_\_construct()** [line 82]

### Constructor for this method

The constructor will setup the required object for this class will initiate the database class, the table class and the template for this class to use

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **See** [advertTemplate::\\$template](#)
- **See** [advertTemplate::\\$table](#)
- **See** db::
- **Access** public

*Integer* function **advertTemplate::create(\$source)** [line 197]

### Function Parameters:

- **Array \$source**

### **This method will take an array and insert it in the database**

This method will insert the formatted information into a database, the format for the array should be an associated array being the first key should be the table inserting with the keys for child array the fields that are being inserted too and the values to insert

```
Array
(
    [users] => Array
        (
            [name] => Dave
            [surname] => Smith
            [email] => dave@dave.com
        )
    [staff] => Array
        (
            [staff_id] => 1245
            [office_number] => 22
            [drown_code] => bee223
        )
)
```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access private**

void function AdvertTemplate::createAdvertTemplateDetails() [[line 605](#)]

### **This method will provide a page to the to add a single row AdvertTemplate to the template table**

The method using the template class to format the information ready to display the the user, so that they may be able to add any of the fields releated to a row in the database. The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function `advertTemplate::deleteAdvertTemplateDetails($id)` [[line 703](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to marked as delete

**Set a row to be marked as deleted**

This method will take the id and set the `delete_date` field to the current datetime, which will marking it as deleted

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function `advertTemplate::editAdvertTemplateDetails($id)` [[line 506](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**Show the details ready to edit of a single AdvertTemplate from the template**

This method is used to display and editable page to the use, so that they maybe able to edit any of the fields releated to the row in question. The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function `advertTemplate::getAdvertTemplateList`([\$type = 'TABLE'], [\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [*line 400*]

**Function Parameters:**

- *String* **\$type** Option of type of response for the output of the list
- *String* **\$orderby** Which single field is used to order the output
- *String* **\$direction** Which direction is required for the orderby output
- *Array* **\$filter** A array of fields to filter, key=TEXT set (eg array('title'=>'this title'))

### Show list of information corresponding the to this class

This Method will produce a list of all the element corresponding to the result of `AdvertTemplate` using the `base/table.class.php` file, which will format the list into a filterable table that uses ajax class to change the content on filtering

There are two response type for this table for the parameter `$type`

TABLE = Will return the content in a table with a filter row and a heading row

AJAX = Will return just the content after evaluating the filter or heading information

The parameter `$filter` expects an array with the key being the field to look for and the value being the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*string* function `advertTemplate::getSelectListOfCategory`(\$id, [\$selectBox = NULL]) [*line 861*]

**Function Parameters:**

- *Integer* **\$id** the id of the row we are looking for
- *Boolean* **\$selectBox** True will return it as a select box with field selected, else just the field

### **This Method will either the catagory\_name feild or a select box of catagory\_name fields**

This Methos is based in the catagory\_id of the category table, it will return either the information in the catagory\_name field as a string or a select box with the id selected

- **Access** public

*string* function advertTemplate::getSelectListOfEmploymentType(\$id, [\$selectBox = NULL]) [*line* [905](#)]

#### **Function Parameters:**

- *Integer* **\$id** the id of the row we are looking for
- *Boolean* **\$selectBox** True will return it as a select box with field selected, else just the field

### **This Method will either the employmenttype feild or a select box of employmenttype fields**

This Methos is based in the employmenttype\_id of the employmenttype table, it will reurn either the information in the employmenttype field as a string or a select box with the id selected

- **Access** public

*string* function advertTemplate::getSelectListOfStates(\$id, [\$selectBox = NULL]) [*line* [883](#)]

#### **Function Parameters:**

- *Integer* **\$id** the id of the row we are looking for
- *Boolean* **\$selectBox** True will return it as a select box with field selected, else just the field

### **This Method will either the name feild or a select box of state fields**

This Methos is based in the state\_id of the state table, it will reurn either the information in the state name field as a string or a select box with the id selected



- **Access public**

*void function* advertTemplate::lists([\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [*line* [115](#)]

**Function Parameters:**

- *String* **\$orderby** Which single field is used to order the output
- *String* **\$direction** Which direction is required for the orderby output
- *Array* **\$filter** A array of fields to filter, key=\$val set (eg array('title'=>'this title'))

**Show will pull a list from the corresponding AdvertTemplate template**

This Method will produce a list of all the element corresponding to the result of AdvertTemplate

I will only pull rows that are not considered delete

eg. the delete\_date field is not "0000-00-00 00:00:00" or set to NULL

The parameter \$filter expects an array with the key being the field to look for and the value being the the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access private**

*void function* advertTemplate::read(\$id) [*line* [242](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will return information as row**

This method is you to get a single row of information from the database based with the primary id and return it as an array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

Boolean function advertTemplate::remove(\$id) [[line 357](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will update a row and make the record as deleted**

This method will take the id and set the delete\_date field to the current datetime, which will marking it as deleted

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

void function advertTemplate::saveAdvertTemplateDetails(\$id) [[line 638](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to updated

**save the information in a single AdvertTemplate to the template table**

This method is used to take the information from createDepartmentDetails and try to validate it through the validation method and on success will format it ready for inserted into the database through the insert method

if the validate fails then the user is show a page that mimics the createDepartmentDetails

method and point out  
error in there input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [advertTemplate::update\(\)](#)
- See [advertTemplate::Validate\(\)](#)
- See [createDepartmentDetails\(\)](#)
- Access public

void function advertTemplate::showAdvertTemplateDetails(\$id) [*line 478*]

**Function Parameters:**

- Integer **\$id** the primary id of the row to show

### Show details of a single AdvertTemplate from the template

This method will return a template page of the information requested  
the method use the template class to format the information ready to display the  
the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- Access public

void function advertTemplate::templateAdvertTemplateLayout(\$fieldMember, [\$input = false], [\$inputArray = array()], \$fielddata) [*line 722*]

**Function Parameters:**

- Array **\$fielddata** An associative array of fields that need to be assigned to the template object
- Boolean **\$input** If false then just assign the value if true the add the value to corresponding form element
- Array **\$inputArray** Not sure :S

- **\$fieldMember**

### **This method assigns the associate array values to the template**

This method is used to incorporate the standards elements of the templates to a single function across all templatd methods

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **TODO** find out what \$inputArray is used for
- **Access** private

Integer function advertTemplate::update(\$source, \$id) [[line 311](#)]

#### **Function Parameters:**

- Array **\$source**
- **\$id**

### **This method will take an array and update a row in the database**

This method will update the formatted information into the database, the format for the array

should be an associated array being the first key should be the table to be updated with the keys

for child array the fields that are being updated too and the values to be updated

```
Array
(
    [users] => Array
        (
            [name] => Dave
            [surname] => Smith
            [email] => dave@dave.com
        )
    [staff] => Array
        (
            [staff_id] => 1245
            [office_number] => 22
            [drown_code] => bee223
        )
)
```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

void function AdvertTemplate::updateAdvertTemplateDetails(\$id) [[line 540](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to updated

**update the information in a single AdvertTemplate from the template**

This method is used to take the information from editDepartmentDetails and try to validate it through the validation method and on success will format it ready for input into the database through the update method

if the validate fails then the user is shown a page that mimics the editDepartmentDetails method and points out error in their input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **See** [AdvertTemplate::update\(\)](#)
- **See** [AdvertTemplate::Validate\(\)](#)
- **See** editDepartmentDetails()
- **Access** public

void function AdvertTemplate::Validate(\$request) [[line 769](#)]

**Function Parameters:**

- *Array* **\$request**

### This method is used to validate inputs from form information

This method will first check the if the fields are in the valid\_field array and strip out any that are not

Then it check that fields that require a value in them from the fields\_required have a value, if not add an error to validation\_error array

Then it will check all the values to find out if the value match the type found in the fields\_validation\_type array, if not add an error to validation\_error array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [advertTemplate::\\$validation\\_error](#)
- See [advertTemplate::\\$fields](#)
- See [advertTemplate::\\$fields\\_validation\\_type](#)
- See [advertTemplate::\\$fields\\_required](#)
- Access public

## Class category

[line [15](#)]

### Category Class

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div> </pre>

- **Package** PeopleScope
- **Author** Jennifer Erator < [jason@lexxcom.com](mailto:jason@lexxcom.com)>
- **Version** 1.1 of the Framework generator

category::\$db

Object = [line [27](#)]

## Database class object

- **Access** private

### **category::\$db\_connect**

*Object = [line [21](#)]*

## **Connect to PDO object through database class**

- **Access** private

### **category::\$fields**

*Array = array('catagory\_id', 'catagory\_name', 'create\_date', 'modify\_date', 'delete\_date') [line [45](#)]*

## **Array of field used in the database if not in this list is dropped from insert or update**

- **Access** private

### **category::\$fields\_required**

*Array|null = array('catagory\_name') [line [51](#)]*

## **Array of feilds require information when validating**

- **Access** private

### **category::\$fields\_validation\_type**

*Array|null* = array ('category\_id'=>'INT', 'category\_name'=>'TEXT', 'create\_date'=>'TEXT', 'modify\_date'=>'TEXT', 'delete\_date'=>'TEXT') [[line 57](#)]

### **Array of feilds and there types that are check when validating**

- **Access** private

### **category::\$table**

*Object* = [[line 33](#)]

### **Table class object**

- **Access** public

### **category::\$template**

*Object* = [[line 39](#)]

### **Template class object**

- **Access** public

### **category::\$validation\_error**

*Array* = array() [[line 64](#)]

### **Array use to store any error found during Validation function**



- **See** Validation()
- **Access** private

Constructor *void* function category::\_\_construct() [*line 82*]

### **Constructor for this method**

The constructor will setup the required object for this class will initiate the database class, the table class and the template for this class to use

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **See** [category::\\$template](#)
- **See** [category::\\$table](#)
- **See** db::
- **Access** public

Integer function category::create(\$source) [*line 176*]

### **Function Parameters:**

- *Array* **\$source**

### **This method will take an array and insert it in the database**

This method will insert the formatted information into a database, the format for the array should be an associated array being the first key should be the table inserting with the keys for child array the fields that are being inserted too and the values to insert

```
Array
(
    [users] => Array
        (
            [name] => Dave
            [surname] => Smith
            [email] => dave@dave.com
        )
    [staff] => Array
        (
            [staff_id] => 1245
        )
)
```

```

        [office_number] => 22
        [drown_code] => bee223
    )
)

```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*void* function category::createCategoryDetails() [*line* [528](#)]

**This method will provide a page to the to add a single row Category to the category table**

The method using the template class to format the information ready to display the the user, so that they may be able to add any of the fields releated to a row in the database. The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function category::deleteCategoryDetails(\$id) [*line* [612](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to marked as delete

**Set a row to be marked as deleted**

This method will take the id and set the delete\_date field to the current datetime, which will marking it as deleted

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function category::editCategoryDetails(\$id) [[line 443](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

### Show the details ready to edit of a single Category from the category

This method is used to display and editable page to the use, so that they maybe able to edit any of the fields releated to the row in question. The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function category::getCategoryList([\$type = 'TABLE'], [\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [[line 365](#)]

**Function Parameters:**

- *String* **\$type** Option of type of response for the output of the list
- *String* **\$orderby** Which single field is used to oder the output
- *String* **\$direction** Which direction os required for the orderby output
- *Array* **\$filter** A array of fields to filter, key=TEXT set (eg array('tile'=>'this title'))

### Show list of information corresponding the to this class

This Method will produce a list of all the element corresponding to the result of Category using the base/table.class.php file, which will format the list into a filtable table that uses ajax class to change the content on filtering

There are two response types for this table for the parameter \$type

TABLE = Will return the content in a table with a filter row and a heading row

AJAX = Will return just the content after evaluating the filter or heading information

The parameter \$filter expects an array with the key being the field to look for and the value being the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void function category::lists([\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [line 115]*

**Function Parameters:**

- **String \$orderby** Which single field is used to order the output
- **String \$direction** Which direction is required for the orderby output
- **Array \$filter** An array of fields to filter, key=\$val set (eg array('title'=>'this title'))

### **Show will pull a list from the corresponding Category category**

This Method will produce a list of all the elements corresponding to the result of Category

I will only pull rows that are not considered deleted

eg. the delete\_date field is not "0000-00-00 00:00:00" or set to NULL

The parameter \$filter expects an array with the key being the field to look for and the value being the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*void* function category::read(\$id) [[line 221](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will return information as row**

This method is you to get a single row of information from the database based ith the primary id and return it as an array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*Boolean* function category::remove(\$id) [[line 322](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will update a row and make the recored as deleted**

This method will take the id and set the delete\_date field to the current datetime, which will marking it as deleted

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*void* function category::saveCategoryDetails(\$id) [[line 561](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to updated

### save the information in a single Category to the category table

This method is used to take the information from createDepartmentDetails and try to validate it through the validation method and on success will format it ready for inserted into the database through the insert method

if the validate fails then the user is show a page that mimics the createDepartmentDetails method and point out error in there input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [category::update\(\)](#)
- See [category::Validate\(\)](#)
- See createDepartmentDetails()
- Access public

void function category::showCategoryDetails(\$id) [[line 415](#)]

#### **Function Parameters:**

- *Integer* **\$id** the primary id of the row to show

### Show details of a single Category from the category

This method will return a template page of the information requested the method use the template class to format the information ready to display the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- Access public

void function category::templateCategoryLayout(\$fieldMember, [\$input = false], [\$inputArray = array()], \$fielddata)  
[line 631]

**Function Parameters:**

- **Array \$fielddata** An associative array of fields that need to be assigned to the template object
- **Boolean \$input** If false then just assign the value if true the add the value to corresponding form element
- **Array \$inputArray** Not sure :S
- **\$fieldMember**

**This method assigns the associate array values to the template**

This method is used to incorporate the standards elements of the templates to a single function across all tempatled methods

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **TODO** find out what \$inputArray is used for
- **Access** private

Integer function category::update(\$source, \$id) [line 276]

**Function Parameters:**

- **Array \$source**
- **\$id**

**This method will take an array and update a row in the database**

This method will update the formatted information into the database, the format for the array

should be an associated array being the first key should be the table to be updated with the keys

for child array the fields that are being updated too and the values to be updated

```
Array  
(  
    [users] => Array
```

```
(
    [name] => Dave
    [surname] => Smith
    [email] => dave@dave.com
)
[staff] => Array
(
    [staff_id] => 1245
    [office_number] => 22
    [drown_code] => bee223
)
)
```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

void function category::updateCategoryDetails(\$id) [[line 477](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to updated

### update the information in a single Category from the category

This method is used to take the information from editDepartmentDetails and try to validate it thought the validation method and on success will format it ready for input into the database through the update method

if the validate fails then the user is show a page that mimics the editDepartmentDetails method and point out error in there input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>



- See [category::update\(\)](#)
- See [category::Validate\(\)](#)
- See [editDepartmentDetails\(\)](#)
- Access public

void function category::Validate(\$request) [*line 665*]

**Function Parameters:**

- Array \$request

**This method is used to validate inputs from form information**

This method will first check the if the fields are in the valid\_field array and strip out any that are not

Then it check that fields that require a value in them from the fields\_required have a value, if not add an error to validation\_error array

Then it will check all the values to find out if the value match the type found in the fields\_validation\_type array, if not add an error to validation\_error array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [category::\\$validation\\_error](#)
- See [category::\\$fields](#)
- See [category::\\$fields\\_validation\\_type](#)
- See [category::\\$fields\\_required](#)
- Access public

## Class employmenttype

[*line 15*]

### Employmenttype Class

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div> </pre>

- **Package** PeopleScope
- **Author** Jennifer Erator < [jason@lexxcom.com](mailto:jason@lexxcom.com)>
- **Version** 1.1 of the Framework generator

#### **employmenttype::\$db**

*Object = [line [27](#)]*

#### **Database class object**

- **Access** private

#### **employmenttype::\$db\_connect**

*Object = [line [21](#)]*

#### **Connect to PDO object through database class**

- **Access** private

#### **employmenttype::\$fields**

*Array = array('employmenttype\_id', 'employmenttype', 'create\_date', 'modify\_date', 'delete\_date') [line [45](#)]*

#### **Array of field used in the database if not in this list is dropped from insert or update**

- **Access** private

#### **employmenttype::\$fields\_required**

*Array|null = array('employmenttype') [line 51]*

## Array of feilds require information when validating

- **Access** private

### employmenttype::\$fields\_validation\_type

*Array|null = array ('employmenttype\_id'=>'INT', 'employmenttype'=>'TEXT', 'create\_date'=>'TEXT', 'modify\_date'=>'TEXT', 'delete\_date'=>'TEXT') [line 57]*

## Array of feilds and there types that are check when validating

- **Access** private

### employmenttype::\$table

*Object = [line 33]*

## Table class object

- **Access** public

### employmenttype::\$template

*Object = [line 39]*

## Template class object

- **Access** public

**employmenttype::\$validation\_error**

*Array = array() [line 64]*

**Array use to store any error found during Validation function**

- **See** Validation()
- **Access** private

Constructor *void* function employmenttype::\_\_construct() [line 82]

#### **Constructor for this method**

The constructor will setup the required object for this class will initiate the database class, the table class and the template for this class to use

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **See** [employmenttype::\\$template](#)
- **See** [employmenttype::\\$table](#)
- **See** db::
- **Access** public

*Integer* function employmenttype::create(\$source) [line 176]

#### **Function Parameters:**

- *Array* **\$source**

#### **This method will take an array and insert it in the database**

This method will insert the formatted information into a database, the format for the array should be an associated array being the first key should be the table inserting with the keys for child array the fields that are being inserted too and the values to insert

```

Array
(
    [users] => Array
        (
            [name] => Dave
            [surname] => Smith
            [email] => dave@dave.com
        )
    [staff] => Array
        (
            [staff_id] => 1245
            [office_number] => 22
            [drown_code] => bee223
        )
)

```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*void function employmenttype::createEmploymenttypeDetails() [line [528](#)]*

**This method will provide a page to the to add a single row Employmenttype to the employmenttype table**

The method using the template class to format the information ready to display the the user, so that they may be able to add any of the fields releated to a row in the database. The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void function employmenttype::deleteEmploymenttypeDetails(\$id) [line [612](#)]*

### **Function Parameters:**

- *Integer* **\$id** The primary id of the row to marked as delete

### **Set a row to be marked as deleted**

This method will take the id and set the delete\_date field to the current datetime, which will marking it as deleted

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function employmenttype::editEmploymenttypeDetails(\$id) [*line* [443](#)]

### **Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

### **Show the details ready to edit of a single Employmenttype from the employmenttype**

This method is used to display and editable page to the use, so that they maybe able to edit any of the fields releated to the row in question. The method uses the template class to format the information ready to display the the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function employmenttype::getEmploymenttypeList([\$type = 'TABLE'], [\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [*line* [365](#)]

### **Function Parameters:**

- **String \$type** Option of type of response for the output of the list
- **String \$orderby** Which single field is used to order the output
- **String \$direction** Which direction is required for the orderby output
- **Array \$filter** A array of fields to filter, key=TEXT set (eg array('title'=>'this title'))

### Show list of information corresponding to this class

This Method will produce a list of all the element corresponding to the result of Employmenttype using the base/table.class.php file, which will format the list into a filterable table that uses ajax class to change the content on filtering

There are two response types for this table for the parameter \$type

TABLE = Will return the content in a table with a filter row and a heading row  
AJAX = Will return just the content after evaluating the filter or heading information

The parameter \$filter expects an array with the key being the field to look for and the value being the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access public**

void function employmenttype::lists([\$orderby = NULL], [\$direction = 'ASC'], [\$filter = NULL]) [[line 115](#)]

#### Function Parameters:

- **String \$orderby** Which single field is used to order the output
- **String \$direction** Which direction is required for the orderby output
- **Array \$filter** A array of fields to filter, key=\$val set (eg array('title'=>'this title'))

### Show will pull a list from the corresponding Employmenttype employmenttype

This Method will produce a list of all the element corresponding to the result of Employmenttype

I will only pull rows that are not considered delete

eg. the delete\_date field is not "0000-00-00 00:00:00" or set to NULL

The parameter \$filter expects an array with the key being the field to look for and the value being the the information to filter on

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*void* function employmenttype::read(\$id) [[line 221](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will return information as row**

This method is you to get a single row of information from the database based ith the primary id and return it as an array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

*Boolean* function employmenttype::remove(\$id) [[line 322](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to show

**This method will update a row and make the recored as deleted**

This method will take the id and set the delete\_date field to the current datetime, which will marking it as deleted



<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

void function employmenttype::saveEmploymenttypeDetails(\$id) [[line 561](#)]

**Function Parameters:**

- *Integer* **\$id** The primary id of the row to updated

**save the information in a single Employmenttype to the employmenttype table**

This method is used to take the information from createDepartmentDetails and try to validate it through the validation method and on success will format it ready for inserted into the database through the insert method

if the validate fails then the user is show a page that mimics the createDepartmentDetails method and point out error in there input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **See** [employmenttype::update\(\)](#)
- **See** [employmenttype::Validate\(\)](#)
- **See** createDepartmentDetails()
- **Access** public

void function employmenttype::showEmploymenttypeDetails(\$id) [[line 415](#)]

**Function Parameters:**

- *Integer* **\$id** the primary id of the row to show

## Show details of a single Employmenttype from the employmenttype

This method will return a template page of the information requested the method use the template class to format the information ready to display the user

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** public

*void* function employmenttype::templateEmploymenttypeLayout(\$fieldMember, [\$input = false], [\$inputArray = array()], \$fielddata) [*line* [631](#)]

### **Function Parameters:**

- **Array \$fielddata** An associative array of fields that need to be assigned to the template object
- **Boolean \$input** If false then just assign the value if true the add the value to corresponding form element
- **Array \$inputArray** Not sure :S
- **\$fieldMember**

## This method assigns the associate array values to the template

This method is used to incorporate the standards elements of the templates to a single function across all temaptled methods

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **TODO** find out what \$inputArray is used for
- **Access** private

*Integer* function employmenttype::update(\$source, \$id) [*line* [276](#)]

### **Function Parameters:**

- *Array* **\$source**
- **\$id**

### **This method will take an array and update a row in the database**

This method will update the formatted information into the database, the format for the array

should be an associated array being the first key should be the table to be updated with the keys

for child array the fields that are being updated too and the values to be updated

Array

```
(
  [users] => Array
  (
    [name] => Dave
    [surname] => Smith
    [email] => dave@dave.com
  )
  [staff] => Array
  (
    [staff_id] => 1245
    [office_number] => 22
    [drown_code] => bee223
  )
)
```

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- **Access** private

void function employmenttype::updateEmploymenttypeDetails(\$id) [[line 477](#)]

#### **Function Parameters:**

- *Integer* **\$id** The primary id of the row to updated

### **update the information in a single Employmenttype from the employmenttype**

This method is used to take the information from editDepartmentDetails and try to validate it thought the

validation method and on success will format it ready for input into the database through the update method

if the validate fails then the user is show a page that mimics the editDepartmentDetails method and point out error in there input

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [employmenttype::update\(\)](#)
- See [employmenttype::Validate\(\)](#)
- See [editDepartmentDetails\(\)](#)
- Access public

void function employmenttype::Validate(\$request) [*line 665*]

**Function Parameters:**

- Array **\$request**

**This medthod is used to validate inputs from form information**

This method will first check the if the fields are in the valid\_field array and strip out any that are not

Then it check that fields that require a value in them from the fields\_required have a value, if not add an error to validation\_error array

Then it will check all the values to find out if the value match the type found in the fields\_validation\_type array, if not add an error to validation\_error array

<div style="color:red">NOTE: This is generated information from the framework and will need to be corrected if there are any changes</div>

- See [employmenttype::\\$validation\\_error](#)
- See [employmenttype::\\$fields](#)
- See [employmenttype::\\$fields\\_validation\\_type](#)
- See [employmenttype::\\$fields\\_required](#)
- Access public



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

require\_once ['errorhandler.class.php'](#) *line 65*

## required error handler

require\_once ['tools.function.php'](#) *line 61*

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

require\_once '**Mail/mime.php**' [*line [21](#)*]

## Pear Mime class

require\_once '**Mail.php**' [*line [17](#)*]

## Pear Mail class

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



## table.class.php

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

# 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](mailto:jason@lexxcom.com.au)>
- **Version** 1.0
- **Filesource** [Source Code for this file](#)

*void* function Box(\$content, \$header, \$discription) [*line* [421](#)]

### **Function Parameters:**

- **\$content**
- **\$header**
- **\$discription**

*Integer* function convertDecimalToMinutes(\$percent) [*line* [374](#)]

### **Function Parameters:**

- *Integer* **\$percent**

**Will take in a percent amount and return values in mins**

eg. 50% = 30mins

*String* function convertMinutes2Hours(\$Minutes) [*line* [389](#)]

### **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 [301](#)]

**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 [196](#)]

**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 [343](#)]

**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 [359](#)]

**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 [137](#)]

**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 [259](#)]

**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 [236](#)]

**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 161]

**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 84]

**Output \$GLOBAL['trace'] Trace**

String function `showVars()` [line 94]

**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 178]`

**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

`void function warning($content) [line 438]`

**Function Parameters:**

- **\$content**

**\$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 [46](#)]

**Generate a formatted exception error**

- **Access** public

*void* function CustomException::messageError() [line [36](#)]

- **Access** public

*void* function CustomException::queryError(\$query) [line [72](#)]

**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 74]

**This class is Database abstraction layer**

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

**db::\$dbh**

*mixed* = [line 77]

**db::\$dsn**

*mixed* = [line 76]

**db::\$lastQuery**

*mixed* = [line 78]

Constructor *Void* function db::\_\_construct() [line 86]

**constructor to initiate database conection**

Array function db::delete(\$sql) [line 151]

**Function Parameters:**

- *string* **\$sql**

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



- **Access** public

*String* function db::getDNSString() [[line 105](#)]

**returns current DSN string used to connect ot the server**

- **Access** public

*Array* function db::insert(\$sql, [\$exec = NULL]) [[line 133](#)]

**Function Parameters:**

- *String* **\$sql**
- **\$exec**

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

- **Access** public

*string* function db::prepareToQuery(\$query, \$params) [[line 230](#)]

**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, [\$exec = NULL]) [[line 185](#)]

**Function Parameters:**

- *string* **\$sql**
- **\$exec**

**Will instigate a query and return a recordset;**

- **Access** public

Array function db::select(\$sql) [[line 115](#)]

**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, [\$exec = NULL]) [[line 168](#)]

**Function Parameters:**

- *string* **\$sql**
- **\$exec**

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

- **Access** public

## Class email

[line [29](#)]

This class is used create and send email

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

**email::\$crlf**

*String* = "\n" [line [35](#)]

**Carrage return**

**email::\$header**

*Araay* = [line [59](#)]

**Mail Header information**

**email::\$mail**

*Object* = [line [47](#)]

**Mail Object**

**email::\$mime**

*unknown\_type* = [\[line 41\]](#)

## Set Mime Type

**email::\$template**

*Object* = [\[line 53\]](#)

## Template Object

Constructor *void* function email::email() [\[line 74\]](#)

## Constructor for Email php4

*void* function email::append\_file(\$file, \$type) [\[line 103\]](#)

### **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) [\[line 84\]](#)

### **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) [\[line 93\]](#)

### **Function Parameters:**

- *String* **\$text**

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

*void* function email::recordEmail(\$stable, \$content, [\$form\_name = NULL]) [\[line 196\]](#)

### **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* [232](#)]

### **Function Parameters:**

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

## compile and Send email

*void* function email::sender(\$from) [*line* [160](#)]

### **Function Parameters:**

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

## Set the Sender info for header

*void* function email::setHeader(\$header) [*line* [178](#)]

### **Function Parameters:**

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

## Set all other Header information as required

*void* function email::setHTMLtemplate(\$template, \$content) [*line* [114](#)]

### **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 129]`

**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 168]`

**Function Parameters:**

- *String* **\$subject**

**Set the Subject line fo the email**

`void function email::_construct($type) [line 66]`

**Function Parameters:**

- *unknown\_type* **\$type**

**Constructor for Email php5**

# Class form

[line 43]

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

- **Package** PeopleScope
- **Sub-Package** Base
- **Author** Jason Stewart < [jason@lexxcom.com.au](mailto:jason@lexxcom.com.au)>
- **Version** 1.0

form::\$autoRefresh

*String* = [line 88]

JavaScript handle onChange for Form input types command

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

form::\$autoRefreshcheckboxs

*String* = [line 96]

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 104]

## 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 49]*

**Database object**

**form::\$enctype**

*String = [line 61]*

**Form enctype type**

**form::\$fck\_configUrl**

*String = [line 110]*

**Url Tor find fck config file**

**form::\$fck\_height**

*Integer = 500 [line 79]*

**Height size of fck editor**

**form::\$fck\_width**

*Integer = 300 [line 73]*

**Width size of fck editor**

**form::\$form**

*String = "" [line 55]*

**Holds string value for form builder**



**form::\$formScript**

*String* = [line [67](#)]

## Java scripts reloaded to this class

Constructor *void* function form::\_\_construct(\$form) [line [118](#)]

### **Function Parameters:**

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

## Consrtructor

Constructor *void* function form::form(\$form) [line [128](#)]

### **Function Parameters:**

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

## Consrtructor php4

*String* function form::draw([\$column = NULL]) [line [213](#)]

### **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 [199](#)]

### **Setup closing form tag**

*string* function form::formHeader(\$action, [\$method = NULL], [\$formname = NULL]) [line [164](#)]

### **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* [143](#)]

### **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* [152](#)]

### **Function Parameters:**

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

## Standards Submit Button

*Html* function form::inputfield(\$input) [*line* [255](#)]

### **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* [335](#)]

### **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* [244](#)]

### **Function Parameters:**

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

## Set a alternate FCK config file path

*void* function form::unsetAutoRefresh() [*line* [347](#)]

## Remove the AutoRefresh function command

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

## Class table

[*line* [34](#)]

### 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
- **Sub-Package** Base
- **Author** Jason Stewart < [jason@lexxcom.com.au](mailto:jason@lexxcom.com.au) >
- **Version** 1.0

#### **table::\$basePage**

*mixed* = NULL [line [43](#)]

- **Access** private

#### **table::\$columnsWidth**

*mixed* = array() [line [40](#)]

- **Access** private

#### **table::\$filterArray**

*mixed* = array() [line [38](#)]

- **Access** private

#### **table::\$footer**

*mixed* = [line [48](#)]

- **Access** private

#### **table::\$headerArray**

*mixed* = array() [line [37](#)]

- **Access** private

#### **table::\$id**

*mixed* = [line [41](#)]

- **Access** private

#### **table::\$identifier**

*mixed* = [line [36](#)]

- **Access** private

#### **table::\$link\_action**

*mixed* = 'show' [[line 46](#)]

- **Access** private

#### **table::\$link\_field**

*mixed* = [[line 47](#)]

- **Access** private

#### **table::\$name**

*mixed* = 'list' [[line 42](#)]

- **Access** private

#### **table::\$nolink**

*mixed* = false [[line 49](#)]

- **Access** private

#### **table::\$removeColumnArray**

*mixed* = array() [[line 39](#)]

- **Access** private

#### **table::\$rowsOnly**

*mixed* = false [[line 50](#)]

- **Access** private

**table::\$row\_class\_field\_name**

*mixed* = [[line 45](#)]

- **Access** private

**table::\$row\_class\_name**

*mixed* = [[line 44](#)]

- **Access** private

**table::\$SEOurl**

*mixed* = SEO\_LINK [[line 51](#)]

- **Access** private

Constructor *void* function table::\_\_construct([*\$id* = NULL]) [[line 54](#)]

**Function Parameters:**

- ***\$id***

*void* function table::buildTd(*\$key*, *\$value*) [[line 346](#)]

**Function Parameters:**

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

- **Access** private

*void* function table::buildWhereArrayFromRequest([\$full\_like = NULL]) [*line* [369](#)]

**Function Parameters:**

- **\$full\_like**

- **Access** public

*void* function table::generateDisplayTable(\$content) [*line* [146](#)]

**Function Parameters:**

- **\$content**

- **Access** public

*void* function table::getFilterType(\$type, \$key, [\$content = NULL]) [*line* [299](#)]

**Function Parameters:**

- **\$type**
- **\$key**
- **\$content**



- **Access** private

*void* function table::getOrderType(\$type, \$key) [*line* [339](#)]

***Function Parameters:***

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

- **Access** private

*void* function table::removeColumn(\$columnArray) [*line* [85](#)]

***Function Parameters:***

- **\$columnArray**

- **Access** public

*void* function table::setBasePage(\$basepage) [*line* [125](#)]

***Function Parameters:***

- **\$basepage**

- **Access** public

*void* function table::setColumnsClass(\$columnClassArray) [*line* [93](#)]

***Function Parameters:***

- **\$columnClassArray**

- **Access** public

*void* function table::setColumnsWidth(\$columnWidthArray) [*line* [89](#)]

***Function Parameters:***

- **\$columnWidthArray**

- **Access** public

*void* function table::setFilter(\$filterArray) [*line* [81](#)]

***Function Parameters:***

- **\$filterArray**

- **Access** public

*void* function table::setFooter(\$field) [*line* [77](#)]

***Function Parameters:***

- **\$field**

- **Access** public

*void* function table::setHeader(\$headerArray) [*line* [69](#)]

***Function Parameters:***

- **\$headerArray**

- **Access** public

*void* function table::setIdentifier(\$field, [\$no\_link = false]) [*line* [97](#)]

***Function Parameters:***

- **\$field**
- **\$no\_link**

- **Access** public

*void* function table::setIdentifierPage(\$page) [*line* [114](#)]

***Function Parameters:***

- **\$page**

- **Access** public

*void* function table::setLinkAction(\$action) [*line* [106](#)]

***Function Parameters:***

- **\$action**

- **Access** public

*void* function table::setLinkField(\$action) [[line 110](#)]

***Function Parameters:***

- **\$action**

- **Access** public

*void* function table::setPrimaryId(\$id) [[line 129](#)]

***Function Parameters:***

- **\$id**

- **Access** public

*void* function table::setRowClassFieldName(\$name) [[line 142](#)]

***Function Parameters:***

- **\$name**

- **Access** public

*void function* table::setRowClassName(\$name) [[line 138](#)]

***Function Parameters:***

- *\$name* **\$name**

## **Set the Class name for a Row in the table**

- **Access** public

*void function* table::setRowsOnly() [[line 73](#)]

- **Access** public

*void function* table::setTableName(\$name) [[line 121](#)]

***Function Parameters:***

- **\$name**

- **Access** public

*void function* table::\_\_destruct() [[line 65](#)]

# 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 replaced 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

# Appendices

# Appendix A - Class Trees

## Package PeopleScope

### advertisement

- [advertisement](#)

### advertTemplate

- [advertTemplate](#)

### category

- [category](#)

### CustomException

- Exception
  - [CustomException](#)

### db

- [db](#)

### email

- [email](#)

## employmenttype

- [employmenttype](#)

## table

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

## template

- [template](#)

Package default

## login

- [login](#)

## template

- [template](#)

# Appendix C - Source Code

# Package PeopleScope

# File Source for template.class.php

Documentation for this file is available at [template.class.php](http://www.phpdoc.org/package/PhpDocumentor)

```
1  <?php
2
3
4  class template{
5
6      private $db_connect;
7      private $db;
8      private $layout = 'layout.tpl.html';
9      private $headerArray;
10     private $filterArray;
11     private $template;
12
13
14
15     public function __construct($layout = NULL){
16
17         $this-> db = new db();
18         //pp($this->db);
19         //$this->db_connect = $this->db->db;
20
21         //if ($this->db->lastError){
22         //    $this->lastError = $this->db->lastError;
23         //    return false;
24         //}
25
26
27         $this-> template = new stdClass();
28
29         if($layout){
30             $this-> template-> layout = fread(fopen( DIR_ROOT."/templates/" . $layout,
31 'r'), filesize(DIR_ROOT."/templates/" . $layout));
32         }else{
33             $this-> template-> layout = fread(fopen(
34 DIR_ROOT."/templates/layout.tpl.html" , 'r'),
35 filesize(DIR_ROOT."/templates/layout.tpl.html" ));
36         }
37
38         if(DEBUG){
39             $this-> template-> layout = str_replace("{*showVars*}" , showVars(),
40 $this-> template-> layout );
41         }
42     }
43
44     public function __destruct(){
45     }
46
47     public function page($field){
48         $val = fread(fopen( DIR_ROOT."/templates/" . $field, 'r'),
49 filesize(DIR_ROOT."/templates/" . $field));
50
51         $this-> template-> layout = str_replace("{*CONTENT*}" , $val, $this-
52 > template-> layout );
53     }
54
55     public function insert($template){
56         $this-> chunk = fread(fopen( DIR_ROOT."/templates/" . $template, 'r'),
57 filesize(DIR_ROOT."/templates/" . $template));
58         return $this-> chunk;
59     }
60
61     public function content($field){
62         $this-> template-> layout = str_replace("{*CONTENT*}" , $field, $this-
63 > template-> layout );
64     }
65 }
```



```

60     public function assign($field, $value, & $tpl=NULL ){
61
62         if($tpl){
63             $tpl = str_replace('{*'.$field.'*}', $value, $tpl );
64         }else{
65             $this-> template-> layout = str_replace('{*'.$field.'*}', $value, $this-
> template-> layout );
66         }
67     }
68
69     public function fetch(& $tpl=NULL){
70         if($tpl){
71             $striped_layout = $this-> strip_tags($tpl);
72         }else{
73             $striped_layout = $this-> strip_tags($this-> template-> layout);
74         }
75         return $striped_layout;
76     }
77
78     public function display(){
79         echo $this-> fetch();
80     }
81
82     public function formatBoolean($value){
83         if ($value){
84             return 'Yes';
85         }else{
86             return 'No';
87         }
88
89         return "<div class=\"error\">error??</div>" ;
90     }
91
92     public function formatValue($value, $msg){
93         if (!empty($value)){
94             return $value;
95         }else{
96             return $msg;
97         }
98
99         return "<div class=\"error\">error??</div>" ;
100     }
101
102     public function getListTable($table, $value, $idField, $valueField, $selectBox = NULL, $WHERE =
NULL){
103
104         //set vars
105         $retValue = '';
106         $selected = '';
107         $other = '';
108
109         $sql = " SELECT * FROM $table " ;
110
111         if(!empty($selectBox)){
112             $sql .= $WHERE;
113             $sql .= " ORDER BY $valueField" ;
114             foreach ($this-> db-> select($sql) as $key=> $row){
115
116                 $selected = ($row[$idField] == $value)? 'SELECTED' : '';
117                 if (trim(strtoupper($row[$valueField])) === "OTHER" ){
118                     $other = "\t<option value=\"" . $row[$idField]. "\"
" . $selected." style=\"background-
color:#efefef\">" . $row[$valueField]. "</option>\n" ;
119                 }else{
120                     $retValue .= "\t<option value=\"" . $row[$idField]. "\"
" . $selected.">" . $row[$valueField]. "</option>\n" ;
121                 }
122             }
123             $retValue .= $other;
124         }else{
125             if (empty($value)){
126                 return ;
127             }
128             $sql .= " WHERE " . $idField." = " . $value." ;
129
130             $a = $this-> db-> select($sql);
131             //$a = $stmt->fetch();
132             $retValue = $a[0][$valueField];
133         }
134     }

```

```

135         return $retValue;
136     }
137
138     public function input($type, $name, $value=NULL){
139
140         switch(strtoupper($type)){
141             CASE 'TEXT' : $retValue = "<input type='text'
name=\"$name\" id=\"$name\" value=\"$value\">";
BREAK;
142             CASE 'PASSWORD' : $retValue = "<input type='password'
name=\"$name\" id=\"$name\" value=\"$value\">";
BREAK;
143             CASE 'HIDDEN' : $retValue = "<input type='hidden'
name=\"$name\" id=\"$name\" value=\"$value\">";
BREAK;
144             CASE 'TEXTAREA' : $retValue = "<textarea name=\"$name\"
id=\"$name\" value=\"$value\">"; BREAK;
145             CASE 'CHECKBOX' : $checked = (strtoupper($value) == 'YES')? "checked
='checked'" : '';
146             $retValue = "<input type='checkbox'
name=\"$name\" id=\"$name\" value='1' \"$checked.\" />"; BREAK;
147             CASE 'RADIO' : $retValue = "<input type='radio'
name=\"$name\" id=\"$name\" value=\"$value\" />"; BREAK;
148         }
149         return $retValue;
150     }
151
152     private function strip_tags($string){
153         preg_match_all("/{\\*(.*)\\*}\\s\\s", $string, $matching_data);
154         return $string = str_replace($matching_data[0], "", $string);
155     }
156
157     public function externalLink($link){
158         if(!empty($link)){
159             return "<a href=\"$link\"
target=\"_blank\">\" \"$link.\"</a>";
160         }
161     }
162     return;
163 }
164
165
166 }

```

# File Source for login.class.php

Documentation for this file is available at [login.class.php](http://login.class.php)

```
1  <?php
2
3  class login {
4
5      public $lastError;
6      public $table;
7      public $template;
8      private $error;
9      public $admin;
10
11     function __construct(){
12         $this-> table = new table();
13         $this-> template = new template();
14     }
15
16     public function getHomePage(){
17         $this-> template = new template();
18         $this-> template-> page('index.tpl.html');
19         echo $this-> template-> display();
20     }
21
22     function checkUserLogin($uname, $password){
23         try {
24             $conn2 = new PDO(MAIN_DB_TYPE.':dbname='.MAIN_DB_DBASE.':host='.MAIN_DB_HOST,
25 MAIN_DB_USER, MAIN_DB_PASS);
26         } catch (PDOException $e) {
27             echo $e-> getMessage();
28         }
29
30         $conn2-> setAttribute (PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
31
32         $sql = "SELECT users.user_id,clients_users.client_id,server_connection.* FROM
33             users
34             Left Join clients_users ON users.user_id = clients_users.user_id
35             Left Join clients_server_connection ON clients_users.client_id =
36             clients_server_connection.client_id
37             Left Join server_connection ON clients_server_connection.connection_id =
38             server_connection.connection_id
39             WHERE user_name='          . $uname. ' AND
40             user_password='          . $password. ' "          ;
41
42         try{
43             $stmt2 = $conn2-> prepare($sql);
44         }catch(CustomException $e){
45             throw new CustomException('QUERY : '. $e-> getMessage());
46         }
47
48         $stmt2-> execute();
49
50         if ($stmt2-> errorCode() != 00000 )
51         {
52             $error = $stmt2-> errorInfo();
53             throw new CustomException($error[2]);
54         }
55
56         $result = $stmt2-> fetch(PDO::FETCH_NAMED);
57
58         $_SESSION['dbaccess']['server_host'] = $result['server_host'];
59         $_SESSION['dbaccess']['server_port'] = $result['server_port'];
60         $_SESSION['dbaccess']['server_database'] = $result['server_database'];
61         $_SESSION['dbaccess']['server_type'] = $result['server_type'];
62         $_SESSION['user']['client_id'] = $result['client_id'];
63         $_SESSION['user']['user_id'] = $result['user_id'];
64
65         pp($_SESSION);
66     }
67 }
```

```
64  
65     }  
66  
67 }
```

## Package PeopleScope

# File Source for advertisement.class.php

Documentation for this file is available at [advertisement.class.php](http://www.phpdoc.org/package/PhpDocumentor/advertisement.class.php)

```
1  <?php
2  /**
3   * Advertisement Class
4   * <pre>
5   * This class is based on the table advertisement
6   *
7   * <div style="color:red">NOTE: This is generated information from the framework
and will need to be corrected if there are any changes</div>
8   * </pre>
9   *
10  * @author Jennifer Erator <jason@lexxcom.com>
11  * @version 1.1 of the Framework generator
12  * @package PeopleScope
13  */
14
15  class advertisement {
16
17      /**
18       * Connect to PDO object through database class
19       * @var Object
20       */
21      private $db_connect;
22
23      /**
24       * Database class object
25       * @var Object
26       */
27      private $db;
28
29      /**
30       * Table class object
31       * @var Object
32       */
33      public $table;
34
35      /**
36       * Template class object
37       * @var Object
38       */
39      public $template;
40
41      /**
42       * Array of field used in the database if not in this list is dropped from insert or update
43       * @var Array
44       */
45      private $fields = array('advertisement_id', 'title', 'catagory_id', 'template_id', 'office_id',
'dept_id', 'role_id', 'state_id', 'store_location_id', 'storerole_id', 'start_date', 'end_date',
'discription', 'requirments', 'upload_resume', 'cover_letter', 'status', 'employmenttype_id',
'create_date', 'create_by', 'modify_date', 'modify_by', 'delete_date', 'delete_by');
46
47      /**
48       * Array of feilds require information when validating
49       * @var Array|null
50       */
51      private $fields_required = array('title');
52
53      /**
54       * Array of feilds and there types that are check when validating
55       * @var Array|null
56       */
57      private $fields_validation_type = array ('advertisement_id'=> 'TEXT', 'title'=> 'TEXT',
'catagory_id'=> 'INT', 'template_id'=> 'INT', 'office_id'=> 'INT', 'dept_id'=> 'INT',
'role_id'=> 'INT', 'state_id'=> 'INT', 'store_location_id'=> 'INT', 'storerole_id'=> 'INT',
'start_date'=> 'DATE', 'end_date'=> 'DATE', 'discription'=> 'TEXT', 'requirments'=> 'TEXT',
'upload_resume'=> 'BOOL', 'cover_letter'=> 'BOOL', 'status'=> 'BOOL',
'employmenttype_id'=> 'INT', 'create_date'=> 'TEXT', 'create_by'=> 'INT', 'modify_date'=> 'TEXT',
'modify_by'=> 'INT', 'delete_date'=> 'TEXT', 'delete_by'=> 'TEXT');
```

```

58
59 /**
60  * Array use to store any error found during Validation function
61  * @see Validation()
62  * @var Array
63  */
64 private $validation_error = array();
65
66 /**
67  * Contructor for this method
68  *
69  * <pre>
70  * The constructor will setup the required object for this class
71  * will initiate the database class, the table class and the template
72  * for this class to use
73  *
74  *
75  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
76  * </pre>
77  *
78  * @see db::
79  * @see table
80  * @see template
81  */
82 public function __construct(){
83     $this-> db = new db();
84
85     try {
86         $this-> db_connect = $this-> db-> dbh;
87     } catch (CustomException $e) {
88         $e-> logError();
89     }
90
91     $this-> table = new table();
92     $this-> template = new template();
93 }
94
95 /**
96  * Show will pull a list from the corresponding Advertisement advertisement
97  *
98  *
99  * <pre>
100  * This Method will produce a list of all the element corresponding to the result of
Advertisement
101  *
102  * I will only pull rows that are not considered delete
103  * eg. the delete_date field is not "0000-00-00 00:00:00" or set to NULL
104  *
105  * The parameter $filter expects an array with the key being the field to look for and the
106  * value being the the information to filter on
107  *
108  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
109  * </pre>
110  *
111  * @param String $orderby Which single field is used to oder the output
112  * @param String $direction Which direction os required for the orderby output
113  * @param Array $filter A array of fields to filter, key=$val set (eg array('tile'=>'this
title'))
114  */
115 Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
116
117     $sql = "SELECT advertisement.advertisement_id,
118             advertisement.title,
119             category.catagory_id,
120             category.catagory_name,
121             template.title AS template_title,
122             advertisement.office_id,
123             advertisement.dept_id,
124             advertisement.role_id,
125             advertisement.state_id,
126             state.name,
127             advertisement.store_location_id,
128             advertisement.storerole_id,
129             DATE_FORMAT(advertisement.start_date, '%d/%m/%Y') AS start_date,
130             DATE_FORMAT(advertisement.end_date, '%d/%m/%Y') AS end_date,
131             advertisement.discription,
132             advertisement.requirments,
133             if(advertisement.upload_resume, 'yes', 'no') AS upload_resume,

```

```

134         if(advertisement.cover_letter, 'yes', 'no') AS cover_letter,
135         advertisement.status,
136         advertisement.employmenttype_id,
137         employmenttype,
138         advertisement.create_date,
139         create_by,
140         advertisement.modify_date,
141         modify_by,
142         advertisement.delete_date,
143         delete_by
144     FROM advertisement
145         LEFT JOIN category ON advertisement.catagory_id = category.catagory_id
146         LEFT JOIN state ON advertisement.state_id = state.state_id
147         LEFT JOIN employmenttype ON advertisement.employmenttype_id =
employmenttype.employmenttype_id
148         LEFT JOIN template ON advertisement.template_id = template.template_id
149     WHERE (advertisement.delete_date = '00-00-0000 00:00:00' OR
advertisement.delete_date IS NULL)"
150     ;
151     if(is_array($filter)){
152         foreach($filter AS $key=> $value){
153             if ($value != 'NULL' && !empty($value)){
154                 $sql .= " AND " . $value;
155             }
156         }
157     }
158     if($orderby){
159         $sql .= " ORDER BY " . $orderby . " " . $direction;
160     }
161     try{
162         $result = $this->db->select($sql);
163     }catch(CustomException $e){
164         echo $e->queryError($sql);
165     }
166     return $result;
167 }
168
169 /**
170  * This method will take an array and insert it in the database
171  *
172  * <pre>
173  * This method will insert the formatted information into a database, the format for the array
174  * should be an associated array being the first key should be the table inserting with the keys
175  * for child array the fields that are being inserted too and the values to insert
176  *
177  * Array
178  * (
179  *     [users] => Array
180  *         (
181  *             [name] => Dave
182  *             [surname] => Smith
183  *             [email] => dave@dave.com
184  *         )
185  *     [staff] => Array
186  *         (
187  *             [staff_id] => 1245
188  *             [office_number] => 22
189  *             [drown_code] => bee223
190  *         )
191  * )
192  *
193  * <div style="color:red">NOTE: This is generated information from the
194  * framework and will need to be corrected if there are any changes</div>
195  *
196  * </pre>
197  *
198  * @param Array $source
199  *
200  * @return Integer Return last inserted primary id
201  */
202 Private function create($source){
203     try{
204         $this->db_connect->beginTransaction();
205
206         foreach($source['advertisement'] AS $key=> $val){
207             $field[] = $key;
208             $value[] = ":" . $key;
209         }
210     }

```

```

211     }
212
213     $sql = "INSERT INTO advertisement ( " . implode(' ', $field) . ") VALUES
214     ( " . implode(' ', $value) . ")";
215
216     foreach($source['advertisement'] AS $key=> $val){
217         $exec[":" . $key] = $val;
218     }
219
220     try{
221         $pid = $this-> db-> insert($sql, $exec);
222     }catch(CustomException $e){
223         throw new CustomException($e-> queryError($sql));
224     }
225     $this-> db_connect-> commit();
226 }
227
228 catch (CustomException $e) {
229     $e-> queryError($sql);
230     $this-> db_connect-> rollBack();
231     return false;
232 }
233
234 return $pid;
235 }
236
237 /**
238  * This method will return information as row
239  *
240  * <pre>
241  * This method is you to get a single row of information from the database
242  * based ith the primary id and return it as an array
243  *
244  * <div style="color:red">NOTE: This is generated information from the
245  * framework and will need to be corrected if there are any changes</div>
246  * </pre>
247  * @param Integer $id The primary id of the row to show
248  */
249 Private function read($id){
250
251     $sql = "SELECT advertisement_id,
252             title,
253             advertisement.catagory_id,
254             catagory_name,
255             template_id,
256             office_id,
257             dept_id,
258             role_id,
259             state_id,
260             store_location_id,
261             storerole_id,
262             DATE_FORMAT(start_date, '%d/%m/%Y') AS start_date,
263             DATE_FORMAT(end_date, '%d/%m/%Y') AS end_date,
264             discription,
265             requirments,
266             upload_resume,
267             cover_letter,
268             status,
269             employmenttype_id,
270             DATE_FORMAT(advertisement.create_date, '%d/%m/%Y') AS create_date,
271             create_by,
272             DATE_FORMAT(advertisement.modify_date, '%d/%m/%Y') AS modify_date,
273             modify_by,
274             DATE_FORMAT(advertisement.delete_date, '%d/%m/%Y') AS delete_date,
275             delete_by
276     FROM advertisement
277     LEFT JOIN category ON advertisement.catagory_id = category.catagory_id
278     WHERE advertisement_id = " . $id . "
279     AND (advertisement.delete_date = '00-00-0000 00:00:00' OR
280     advertisement.delete_date IS NULL)" ;
281
282     $stmt = $this-> db_connect-> prepare($sql);
283     $stmt-> execute();
284
285     try{
286         $result = $this-> db-> select($sql);
287     }catch(CustomException $e){

```



```

288         echo $e-> queryError($sql);
289     }
290
291     return $result[0];
292
293 }
294
295 /**
296  * This method will take an array and update a row in the database
297  *
298  * <pre>
299  * This method will update the formatted information into the database, the format for the array
300  * should be an associated array being the first key should be the table to be updated with the
301  * keys
302  * for child array the fields that are being updated too and the values to be updated
303  *
304  * Array
305  * (
306  *     [users] => Array
307  *         (
308  *             [name] => Dave
309  *             [surname] => Smith
310  *             [email] => dave@dave.com
311  *         )
312  *     [staff] => Array
313  *         (
314  *             [staff_id] => 1245
315  *             [office_number] => 22
316  *             [drown_code] => bee223
317  *         )
318  * )
319  *
320  * <div style="color:red">NOTE: This is generated information from the
321  * framework and will need to be corrected if there are any changes</div>
322  * </pre>
323  *
324  * @param Array $source
325  *
326  * @return Integer Return last inserted primary id
327  */
328 Private function update($source, $id){
329     try{
330         $this-> db_connect-> beginTransaction();
331
332         foreach($source['advertisement'] AS $key=> $val){
333             $field[] = $key." = ".$val;
334         }
335
336         $sql = "UPDATE advertisement SET " . implode(' ', $field) . " WHERE
337 advertisement_id = " . $id;
338
339         foreach($source['advertisement'] AS $key=> $val){
340             $exec["."] = $key . $val;
341         }
342
343         try{
344             $pid = $this-> db-> update($sql, $exec);
345         }catch(CustomException $e){
346             throw new CustomException($e-> queryError($sql));
347         }
348         $this-> db_connect-> commit();
349     }
350
351     catch (CustomException $e) {
352         $e-> queryError($sql);
353         $this-> db_connect-> rollBack();
354         return false;
355     }
356
357     header('Location:advertisement.php?action=show&id=' . $id);
358 }
359
360 /**
361  * This method will update a row and make the record as deleted
362  *
363  * <pre>
364  * This method will take the id and set the delete_date field to

```

```

365      * the current datetime, which will marking it as deleted
366      *
367      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
368      * </pre>
369      *
370      * @param Integer $id The primary id of the row to show
371      * @return Boolean success or failed
372      */
373
374 Private function remove($id){
375     if(empty($id)){
376         return false;
377     }
378
379     $sql = "UPDATE advertisement SET delete_date=NOW(),
deleted_by=" . $_SESSION['user']['user_id'] . " WHERE advertisement_id =" . $id;
380
381     try{
382         $result = $this-> db-> update($sql);
383     }catch(CustomException $e){
384         echo $e-> queryError($sql);
385     }
386     return true;
387 }
388
389
390 /***** END CRUD METHOD*****/
391
392
393 /**
394  * Show list of information corresponding the to this class
395  *
396  * <pre>This Method will produce a list of all the element corresponding to the result of
Advertisement
397  * using the base/table.class.php file, which will format the list into a filtable table that
398  * uses ajax class to change the content on filtering
399  *
400  * There are to response type for this table for the parameter $type
401  *
402  * TABLE = Will return the content in a table with a filter row and a heading row
403  * AJAX = Will return just the content after evaluating the filter or heading infomation
404  *
405  * The parameter $filter expects an array with the key being the field to look for and the
406  * value being the the information to filter on
407  *
408  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
409  * </pre>
410  *
411  * @param String $type Option of type of response for the output of the list
412  * @param String $orderby Which single field is used to oder the output
413  * @param String $direction Which direction os required for the orderby output
414  * @param Array $filter A array of fields to filter, key=INT set (eg array('tile'=>'this
title'))
415  */
416
417 public function getAdvertisementList($type='TABLE',$orderby=NULL, $direction='ASC',
$filter=NULL){
418
419     $result = $this-> lists($orderby, $direction, $filter);
420
421     $this-> table-> removeColumn(array('advertisement_id','catagory_id','office_id',
'dept_id', 'role_id', 'state_id', 'store_location_id','storerole_id', 'create_date', 'discription',
'requirments',
'employmenttype_id','create_date','create_by','modify_date','modify_by','delete_date','delete_by'));
422
423     switch(strtoupper($type)){
424
425         case 'AJAX' : $this-> table-> setRowsOnly();
426                     $this-> table-> setIdentifier('advertisement_id');
427                     $this-> table-> setIdentifierPage('advertisement');
428                     echo $this-> table-> generateDisplayTable($result);
429
430         BREAK;
431         case 'TABLE' :
432         DEFAULT :
433             $this-> table-> setHeader(array(
434                 'title'=> 'Title',
435                 'catagory_name'=> 'Catagory',

```

```

436         'template_title'=> 'Template',
437         'name'=> 'State',
438         'start_date'=> 'Start&nbsp;Date' ,
439         'end_date'=> 'End&nbsp;Date' ,
440         'upload_resume'=> 'Resume',
441         'cover_letter'=> 'Cover',
442         'status'=> 'Status',
443         'employmenttype'=> 'Type'
444     ));
445
446     $this-> table-> setFilter(array(
447         'title'=> 'TEXT',
448         'catagory_name'=> 'COMPILED',
449         'template_title'=> 'TEXT',
450         'name'=> 'COMPILED',
451         'start_date'=> 'DATE',
452         'end_date'=> 'DATE',
453         'upload_resume'=> 'COMPILED',
454         'cover_letter'=> 'COMPILED',
455         'status'=> 'COMPILED',
456         'employmenttype'=> 'COMPILED'
457     ));
458
459     $this-> table-> setColumnsWidth(array(
460         '0'=> '300',
461         '1'=> '300',
462         '2'=> '300',
463         '3'=> '300',
464         '4'=> '10',
465         '5'=> '10',
466         '6'=> '10',
467         '7'=> '10',
468         '8'=> '10',
469         '9'=> '10'
470     ));
471
472     $this-> table-> setIdentifier('advertisement_id');
473
474     $this-> template-> content(Box($this-> table-
475 > generateDisplayTable($result),'Advertisement List', 'Shows the current listings for the
476 Advertisement. To create a new Listing <a href="advertisement.php?action=create">Click
477 Here</a>' ));
478
479     $this-> template-> display();
480 }
481
482 /**
483  * Show details of a single Advertisement from the advertisement
484  *
485  * <pre>This method will return a template page of the information requested
486  * the method use the template class to format the information ready to display the
487  * the user
488  *
489  * <div style="color:red">NOTE: This is generated information from the
490  * framework and will need to be corrected if there are any changes</div>
491  * </pre>
492  * @param Integer $id the primary id of the row to show
493  */
494 public function showAdvertisementDetails($id){
495     $fieldMember = $this-> read($id);
496
497     $this-> template-> page('advertisement.tpl.html');
498
499     $this-> templateAdvertisementLayout($fieldMember);
500
501     //if($this->checkAdminLevel(1)){
502         $this-> template-> assign('FUNCTION', "<div class=\"button\"
503 onclick=\"location.href='advertisement.php?action=edit&id=\" . $id . \"\">Edit</d
504 iv>\"");
505     };
506     //}
507
508     echo $this-> template-> fetch();
509 }
510
511 /**
512  * Show the details ready to edit of a single Advertisement from the advertisement
513  */

```

```

510      * <pre>
511      * This method is used to display and editable page to the use, so that they
512      * maybe able to edit any of the fields releated to the row in question.
513      * The method uses the template class to format the information ready to display the
514      * the user
515      *
516      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
517      * </pre>
518      * @param Integer $id The primary id of the row to show
519      */
520      Public function editAdvertisementDetails($id){
521
522          $fieldMember = $this-> read($id);
523
524          $name = 'editAdvertisement';
525
526          $this-> template-> page('advertisement.tpl.html');
527          $this-> template-> assign('FORM-HEADER', '<form
action="advertisement.php?action=update&id=' . $id . '" method="POST"
name="' . $name . '">' );
528
529          $this-> templateAdvertisementLayout($fieldMember, true);
530
531          $this-> template-> assign('FUNCTION', " <div class=\"button\"
onclick=\"document. $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='advertisement.php?action=show&id= " . $id . "'>Cancel<
/div>" );
532
533          $this-> template-> display();
534      }
535
536      /**
537      * update the information in a single Advertisement from the advertisement
538      *
539      * <pre>
540      * This method is used to take the information from editDepartmentDetails and try to validate
it thought the
541      * validation method and on success will format it ready for input into the database through
the update method
542      *
543      * if the validate fails then the user is show a page that mimics the editDepartmentDetails
method and point out
544      * error in there input
545      *
546      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
547      * </pre>
548      *
549      * @see editDepartmentDetails()
550      * @see Validate()
551      * @see update()
552      * @param Integer $id The primary id of the row to updated
553      */
554      Public function updateAdvertisementDetails($id){
555
556          if ($this-> Validate($_REQUEST)){
557
558              $request = $_REQUEST;
559              $table = 'advertisement';
560
561              $save[$table]['title'] = $request['title'];
562              $save[$table]['catagory_id'] = $request['catagory_id'];
563              $save[$table]['template_id'] = $request['template_id'];
564              /*$save[$table]['office_id'] = $request['office_id'];
565              $save[$table]['dept_id'] = $request['dept_id'];
566              $save[$table]['role_id'] = $request['role_id'];*/
567              $save[$table]['state_id'] = $request['state_id'];
568              /*$save[$table]['store_location_id'] = $request['store_location_id'];
569              $save[$table]['storeroles_id'] = $request['storeroles_id'];*/
570              $save[$table]['start_date'] = formatDateUI($request['start_date']);
571              $save[$table]['end_date'] = formatDateUI($request['end_date']);
572              $save[$table]['discription'] = $request['discription'];
573              $save[$table]['requirments'] = $request['requirments'];
574              $save[$table]['upload_resume'] = $request['upload_resume'];
575              $save[$table]['cover_letter'] = $request['cover_letter'];
576              $save[$table]['status'] = $request['status'];
577              $save[$table]['employmenttype_id'] = $request['employmenttype_id'];
578              $save[$table]['modify_by'] = $_SESSION['user']['user_id'];

```

```

579         $save[$stable]['modify_date'] = date('Y-m-d h:i:s');
580
581     $this-> update($save, $id );
582     header('Location: advertisement.php?action=show&id=' . $id);
583 }else{
584
585     $fieldMember = $this-> valid_field;
586     $error = $this-> validation_error;
587
588     $name = 'editAdvertisement';
589
590     $this-> template-> page('advertisement.tpl.html');
591
592     foreach($error AS $key=> $value){
593         $this-> template-> assign('err_'. $key, "<span
class=\"error\">" . @implode(', ', $error[$key]) . "</span>" );
594     }
595
596     $this-> template-> assign('FORM-HEADER', '<form
action="advertisement.php?action=update&id=' . $id . ' method="POST"
name="' . $name . '">');
597
598     $this-> templateAdvertisementLayout($fieldMember, true);
599
600     //if($this->admin->checkAdminLevel(1)){
601         $this-> template-> assign('FUNCTION', " <div
class=\"button\" onclick=\"document. $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='advertisement.php?action=show&id=
' . $id . '\">Cancel<
/div>" );
602     }
603     //}
604     $this-> template-> assign('FORM-FOOTER', '</form>');
605
606     $this-> template-> display();
607 }
608 }
609
610 /**
611  * This method will provide a page to the to add a single row Advertisement to the
advertisement table
612  *
613  * <pre>
614  * The method using the template class to format the information ready to display the
615  * the user, so that they may be able to add any of the fields related to a row in the
database.
616  * The method uses the template class to format the information ready to display the
617  * the user
618  *
619  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
620  * </pre>
621  */
622 Public function createAdvertisementDetails(){
623
624     $name = 'createAdvertisement';
625
626     $this-> template-> page('advertisement.tpl.html');
627     $this-> template-> assign('FORM-HEADER', '<form
action="advertisement.php?action=save" method="POST" name="' . $name . '">');
628
629     $this-> templateAdvertisementLayout('', true);
630
631     $this-> template-> assign('FUNCTION', " <div class=\"button\"
onclick=\"document. $name.submit(); return false\">Save</div><div
class=\"button\"
onclick=\"location.href='advertisement.php?action=list'\">Cancel</div>
" );
632
633     $this-> template-> display();
634 }
635
636 /**
637  * save the information in a single Advertisement to the advertisement table
638  *
639  * <pre>
640  * This method is used to take the information from createDepartmentDetails and try to validate
it thought the
641  * validation method and on success will format it ready for inserted into the database through
the insert method

```

```

643      *
644      * if the validate fails then the user is show a page that mimics the createDepartmentDetails
method and point out
645      * error in there input
646      *
647      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
648      * </pre>
649      *
650      * @see createDepartmentDetails()
651      * @see Validate()
652      * @see update()
653      * @param Integer $id The primary id of the row to updated
654      */
655      Public function saveAdvertisementDetails(){
656
657          if ($this-> Validate($_REQUEST)){
658
659              $request = $_REQUEST;
660              $table = 'advertisement';
661
662              $save[$table]['title'] = $request['title'];
663              $save[$table]['catagory_id'] = $request['catagory_id'];
664              $save[$table]['template_id'] = $request['template_id'];
665              /*$save[$table]['office_id'] = $request['office_id'];
666              $save[$table]['dept_id'] = $request['dept_id'];
667              $save[$table]['role_id'] = $request['role_id'];*/
668              $save[$table]['state_id'] = $request['state_id'];
669              /*$save[$table]['store_location_id'] = $request['store_location_id'];
670              $save[$table]['storerole_id'] = $request['storerole_id'];*/
671              $save[$table]['start_date'] = formatDateUI($request['start_date']);
672              $save[$table]['end_date'] = formatDateUI($request['end_date']);
673              $save[$table]['discription'] = $request['discription'];
674              $save[$table]['requirments'] = $request['requirments'];
675              $save[$table]['upload_resume'] = $request['upload_resume'];
676              $save[$table]['cover_letter'] = $request['cover_letter'];
677              $save[$table]['status'] = $request['status'];
678              $save[$table]['employmenttype_id'] = $request['employmenttype_id'];
679              $save[$table]['create_by'] = $_SESSION['user']['user_id'];
680
681              $save[$table]['create_date'] = date('Y-m-d h:i:s');
682
683              $id = $this-> create($save);
684              header('Location: advertisement.php?action=show&id=' . $id);
685          }else{
686
687              $fieldMember = $this-> valid_field;
688
689              $error = $this-> validation_error;
690
691              $name = 'createAdvertisement';
692
693              $this-> template-> page('advertisement.tpl.html');
694
695              foreach($error AS $key=> $value){
696                  $this-> template-> assign('err_' . $key, "<span
class=\"error\">" . @implode(' , ' , $value) . "</span>" );
697              }
698
699              $this-> template-> assign('FORM-HEADER', '<form
action="advertisement.php?action=save" method="POST" name="' . $name . '">' );
700
701              $this-> templateAdvertisementLayout($fieldMember, true);
702
703              //if($this->admin->checkAdminLevel(1)){
704                  $this-> template-> assign('FUNCTION', " <div
class=\"button\" onclick=\"document. $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='advertisement.php'>Cancel</div> " );
705              }
706              $this-> template-> assign('FORM-FOOTER', '</form>' );
707
708              $this-> template-> display();
709          }
710      }
711
712      /**
713      * Set a row to be marked as deleted
714      *
715      * <pre>

```

```

716      * This method will take the id and set the delete_date field to
717      * the current datetime, which will marking it as deleted
718      *
719      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
720      * </pre>
721      * @param Integer $id The primary id of the row to marked as delete
722      */
723      Public function deleteAdvertisementDetails($id){
724          $this-> remove($id);
725          header('Location: advertisement.php');
726      }
727
728      /**
729      * This method assigns the associate array values to the template
730      *
731      * <pre>
732      * This method is used to incorporate the standards elements of the templates to a single
733      * function across all tempailed methods
734      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
735      * </pre>
736      * @todo find out what $inputArray is used for
737      *
738      * @param Array $fielddata An associative array of fields that need to be assigned to the
template object
739      * @param Boolean $input If false then just assign the value if true the add the value to
corresponding form element
740      * @param Array $inputArray Not sure :S
741      */
742      private function templateAdvertisementLayout($fieldMember, $input = false, $inputArray=array()
){
743
744          $id = @$fieldMember['advertisement_id'];
745
746          @$this-> template-> assign('title', ($input)? $this-> template-
> input('text', 'title', $fieldMember['title']):$fieldMember['title']);
747          @$this-> template-> assign('catagory_name', ($input)? $this-
> getSelectListOfCategory($fieldMember['catagory_id'], True):$this-
> getSelectListOfCategory($fieldMember['catagory_id']));
748          @$this-> template-> assign('template_title', ($input)? $this-
> getSelectListOfTemplate($fieldMember['template_id'], True):$this-
> getSelectListOfTemplate($fieldMember['template_id']));
749          @$this-> template-> assign('office_id', ($input)? $this-> template-
> input('text', 'office_id', $fieldMember['office_id']):$fieldMember['office_id']);
750          @$this-> template-> assign('dept_id', ($input)? $this-> template-
> input('text', 'dept_id', $fieldMember['dept_id']):$fieldMember['dept_id']);
751          @$this-> template-> assign('role_id', ($input)? $this-> template-
> input('text', 'role_id', $fieldMember['role_id']):$fieldMember['role_id']);
752          @$this-> template-> assign('state_name', ($input)? $this-
> getSelectListOfStates($fieldMember['state_id'], True):$this-
> getSelectListOfStates($fieldMember['state_id']));
753          @$this-> template-> assign('store_location_id', ($input)? $this-> template-
> input('text', 'store_location_id',
$fieldMember['store_location_id']):$fieldMember['store_location_id']);
754          @$this-> template-> assign('storerole_id', ($input)? $this-> template-
> input('text', 'storerole_id', $fieldMember['storerole_id']):$fieldMember['storerole_id']);
755          @$this-> template-> assign('start_date', ($input)? $this-> template-
> input('text', 'start_date', $fieldMember['start_date']):$fieldMember['start_date']);
756          @$this-> template-> assign('end_date', ($input)? $this-> template-
> input('text', 'end_date', $fieldMember['end_date']):$fieldMember['end_date']);
757          @$this-> template-> assign('discription', ($input)? $this-> template-
> input('text', 'discription', $fieldMember['discription']):$fieldMember['discription']);
758          @$this-> template-> assign('requirments', ($input)? $this-> template-
> input('text', 'requirments', $fieldMember['requirments']):$fieldMember['requirments']);
759          @$this-> template-> assign('upload_resume', ($input)? $this-> template-
> input('checkbox', 'upload_resume', $this-> template-
> formatBoolean($fieldMember['upload_resume']):$this-> template-
> formatBoolean($fieldMember['upload_resume'] ));
760          @$this-> template-> assign('cover_letter', ($input)? $this-> template-
> input('checkbox', 'cover_letter', $this-> template-
> formatBoolean($fieldMember['cover_letter']):$this-> template-
> formatBoolean($fieldMember['cover_letter'] ));
761          @$this-> template-> assign('status', ($input)? $this-> template-
> input('checkbox', 'status', $this-> template-> formatBoolean($fieldMember['status']):$this-
> template-> formatBoolean($fieldMember['status'] ));
762          @$this-> template-> assign('employmenttype', ($input)? $this-
> getSelectListOfEmploymentType($fieldMember['employmenttype_id'], True):$this-
> getSelectListOfEmploymentType($fieldMember['employmenttype_id']));
763          @$this-> template-> assign('create_date', $fieldMember['create_date']);

```



```

764         @$this-> template-> assign('create_by', $this->
> getUserById($fieldMember['create_by']));
765         @$this-> template-> assign('modify_date', $fieldMember['modify_date']);
766         @$this-> template-> assign('modify_by', $this->
> getUserById($fieldMember['modify_by']));
767         @$this-> template-> assign('delete_date', $fieldMember['delete_date']);
768         @$this-> template-> assign('delete_by', $this->
> getUserById($fieldMember['delete_by']));
769
770
771         /*if(isset($id)){
772             $this->template->assign('COMMENTS', $this->comment-
>getCommentBox($id, 'advertisement'));
773         }*/
774
775     }
776
777
778     /**
779     * This method is used to validate inputs from form information
780     * <pre>
781     * This method will first check the if the fields are in the valid_field array and strip out
any that are not
782     * Then it check that fields that require a value in them from the fields_required have a
value, if not add an error to validation_error array
783     * Then it will check all the values to find out if the value match the type found in the
fields_validation_type array, if not add an error to validation_error array
784     * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
785     * </pre>
786     *
787     * @see fields
788     * @see fields_required
789     * @see fields_validation_type
790     * @see validation_error
791     *
792     * @param Array $request
793     */
794     public function Validate($request){
795
796         unset($this-> valid_field);
797         unset($this-> validation_error);
798         $isvalid = True;
799
800         $validfields = $this-> fields;
801         $requiredfields = $this-> fields_required;
802         $fieldsvalidationtype = $this-> fields_validation_type;
803
804         foreach ($request AS $key=> $value){ //lets strip put unwanted or security violation
fields
805             if(in_array($key, $validfields)){
806                 $this-> valid_field[$key] = $value; //pure fields
807             }
808         }
809
810         foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxes
811             if(!isset($this-> valid_field[$value])){
812                 $this-> valid_field[$value] = '';
813             }
814         }
815
816         if(count($requiredfields) > 0 ){
817             foreach($requiredfields AS $value){ // lets check all the required fields have a value
818                 if (empty($this-> valid_field[$value]) || $this-> valid_field[$value] ==
'NULL'){
819                     $this-> validation_error[$value][] = 'Field is Required'; //error field
820                     $isvalid = false;
821                 }
822             }
823         }
824
825
826
827         //now lets validate
828         foreach ($this-> valid_field AS $key=> $value){
829             $value = trim($value);
830             if(!empty($value)){ // don't check if empty, already done in required check
831
832                 switch(@$fieldsvalidationtype[$key]){

```



```

833         case 'TEXTAREA': if (strlen($value) > 1024) {
834             $this-> validation_error[$key][] = 'Field longer than 1024
characters';
835             $isvalid = false;
836         } break;
837         case 'TEXT': if (strlen($value) > 1024) {
838             $this-> validation_error[$key][] = 'Field longer than 1024
characters';
839             $isvalid = false;
840         } break;
841         case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
842             $this-> validation_error[$key][] = 'not a valid SAP number';
843             $isvalid = false;
844         } break;
845         case 'DECIMAL': if (!is_numeric($value)) {
846             $this-> validation_error[$key][] = 'Decimal value expected';
847             $isvalid = false;
848         } break;
849         case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"
) && ( $value != 1)) {
850             $this-> validation_error[$key][] = 'Please check';
851             $isvalid = false;
852         } break;
853         case 'INT': if (!is_numeric($value) && $value != 'NULL') {
854             $this-> validation_error[$key][] = 'Numeric value expected';
855             $isvalid = false;
856         } break;
857         case 'DATE': //$date = str_replace('/', '-', $value);
858             //$date = str_replace("\\", '-', $date);
859             @list($day, $month, $year) = explode('/', $value);
860             if(!checkdate($month,$day, $year)){
861                 $this-> validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
862                 $isvalid = false;
863             } break;
864         case 'YEAR': if(!checkYear($value)){
865             $this-> validation_error[$key][] = 'incorrect year
format, expecting yyyy';
866             $isvalid = false;
867         } break;
868     }
869 }
870 }
871 }
872
873 return $isvalid;
874
875 }
876
877 /**
878  * This Method will either the catagory_name feild or a select box of catagory_name fields
879  *
880  * This Methos is based in the catagory_id of the category table, it will return either the
information
881  * in the catagory_name field as a string or a select box with the id selected
882  *
883  * @todo move this to its own class for all classes to use
884  *
885  * @param Integer $id the id of the row we are looking for
886  * @param Boolean $selectBox True will return it as a select box with field selected, else just
the field
887  * @return string
888  */
889 public function getSelectListOfCategory($id, $selectBox=NULL){
890
891     if($selectBox == false){
892         return $this-> template-> getListTable('category', $id, 'catagory_id',
'catagory_name');
893     }else{
894         $select = "<select name=\"catagory_id\">"
;
895         $select .= "<option value=\"NULL\"></option>"
;
896         $select .= $this-> template-> getListTable('category', $id, 'catagory_id',
'catagory_name', $selectBox);
897         $select .= "</select>"
;
898         return $select;
899     }
900 }
901
902 /**
903  * This Method will either the name feild or a select box of state fields

```

```

904      *
905      * This Method is based in the state_id of the state table, it will reurn either the information
906      * in the state name field as a string or a select box with the id selected
907      *
908      * @todo move this to its own class for all classes to use
909      *
910      * @param Integer $id the id of the row we are looking for
911      * @param Boolean $selectBox True will return it as a select box with field selected, else just
the field
912      * @return string
913      */
914      public function getSelectListOfStates($id, $selectBox=NULL){
915
916          if($selectBox == false){
917              return $this-> template-> getListTable('state', $id, 'state_id', 'name');
918          }else{
919              $select = "<select name=\"state_id\">" ;
920              $select .= "<option value=\"NULL\"></option>" ;
921              $select .= $this-> template-> getListTable('state', $id, 'state_id', 'name',
$selectBox);
922              $select .= "</select>" ;
923              return $select;
924          }
925      }
926
927      /**
928      * This Method will either the employmenttype feild or a select box of employmenttype fields
929      *
930      * This Method is based in the employmenttype_id of the employmenttype table, it will reurn
either the information
931      * in the employmenttype field as a string or a select box with the id selected
932      *
933      * @todo move this to its own class for all classes to use
934      *
935      * @param Integer $id the id of the row we are looking for
936      * @param Boolean $selectBox True will return it as a select box with field selected, else just
the field
937      * @return string
938      */
939      public function getSelectListOfEmploymentType($id, $selectBox=NULL){
940
941          if($selectBox == false){
942              return $this-> template-> getListTable('employmenttype', $id,
'employmenttype_id', 'employmenttype');
943          }else{
944              $select = "<select name=\"employmenttype_id\">" ;
945              $select .= "<option value=\"NULL\"></option>" ;
946              $select .= $this-> template-> getListTable('employmenttype', $id,
'employmenttype_id', 'employmenttype', $selectBox);
947              $select .= "</select>" ;
948              return $select;
949          }
950      }
951
952      /**
953      * This Method will either the employmenttype feild or a select box of employmenttype fields
954      *
955      * This Method is based in the employmenttype_id of the employmenttype table, it will reurn
either the information
956      * in the employmenttype field as a string or a select box with the id selected
957      *
958      * @todo move this as a function in advertTemplate class
959      * @todo move this to its own class for all classes to use
960      *
961      * @param Integer $id the id of the row we are looking for
962      * @param Boolean $selectBox True will return it as a select box with field selected, else just
the field
963      * @return string
964      */
965      public function getSelectListOfTemplate($id, $selectBox=NULL){
966
967          if(is_nan($id) || empty($id)){
968              return;
969          }
970
971          $sql = "Select * from template" ;
972
973          $stmt = $this-> db_connect-> prepare($sql);
974          $stmt-> execute();
975

```

```

976         try{
977             $result = $this-> db-> select($sql);
978         }catch(CustomException $e){
979             echo $e-> queryError($sql);
980         }
981
982         $html = '';
983
984         foreach($result AS $values){
985             if($values['template_id'] == $id){
986                 $html .= '<div class="template-list template-selected"><a
href="advertTemplate.php?action=show&id=' . $values['template_id'] . '">' . $values['title'] . '&
lt;/a></div>'
;
987             }else{
988                 $html .= '<div class="template-list"><a
href="advertTemplate.php?action=show&id=' . $values['template_id'] . '">' . $values['title'] . '&
lt;/a></div>'
;
989             }
990         }
991         return $html;
992     }
993
994     /**
995      * Will get name of the user based on id
996      *
997      * This will return the name and surname concatenated of a user by their id
998      *
999      * @param Integer $id
1000      * @return String The user name
1001      */
1002     public function getUserById($id = false){
1003
1004         if(is_nan($id) || empty($id)){
1005             return;
1006         }
1007
1008         $sql = "      Select CONCAT(name, ' ', surname) as admin from users WHERE user_id =
$id;"
;
1009
1010         $stmt = $this-> db_connect-> prepare($sql);
1011         $stmt-> execute();
1012
1013         try{
1014             $result = $this-> db-> select($sql);
1015         }catch(CustomException $e){
1016             echo $e-> queryError($sql);
1017         }
1018         return $result[0]['admin'];
1019     }
1020 }
1021
1022 }

```

# File Source for advertTemplate.class.php

Documentation for this file is available at [advertTemplate.class.php](#)

```
1  <?php
2  /**
3   * AdvertTemplate Class
4   * <pre>
5   * This class is based on the table template
6   *
7   * <div style="color:red">NOTE: This is generated information from the framework
and will need to be corrected if there are any changes</div>
8   * </pre>
9   *
10  * @author Jennifer Erator <jason@lexxcom.com>
11  * @version 1.1 of the Framework generator
12  * @package PeopleScope
13  */
14
15  class advertTemplate {
16
17      /**
18       * Connect to PDO object through database class
19       * @var Object
20       */
21      private $db_connect;
22
23      /**
24       * Database class object
25       * @var Object
26       */
27      private $db;
28
29      /**
30       * Table class object
31       * @var Object
32       */
33      public $table;
34
35      /**
36       * Template class object
37       * @var Object
38       */
39      public $template;
40
41      /**
42       * Array of field used in the database if not in this list is dropped from insert or update
43       * @var Array
44       */
45      private $fields =array('template_id', 'title', 'employmenttype_id', 'catagory_id', 'office_id',
'dept_id', 'role_id', 'state_id', 'storeLoc_id', 'start_date', 'end_date', 'discription', 'requirments',
'status', 'tracking_id', 'advertisement_id', 'create_date', 'modify_date', 'delete_date');
46
47      /**
48       * Array of feilds require information when validating
49       * @var Array|null
50       */
51      private $fields_required = array('title', 'catagory_id');
52
53      /**
54       * Array of feilds and there types that are check when validating
55       * @var Array|null
56       */
57      private $fields_validation_type = array ('template_id'=> 'TEXT', 'title'=> 'TEXT',
'employmenttype_id'=> 'INT', 'catagory_id'=> 'INT', 'office_id'=> 'INT', 'dept_id'=> 'INT',
'role_id'=> 'INT', 'state_id'=> 'INT', 'storeLoc_id'=> 'INT', 'start_date'=> 'DATE',
'end_date'=> 'DATE', 'discription'=> 'TEXT', 'requirments'=> 'TEXT', 'status'=> 'TEXT',
'tracking_id'=> 'INT', 'advertisement_id'=> 'INT', 'create_date'=> 'TEXT',
'modify_date'=> 'TEXT', 'delete_date'=> 'TEXT');
58
59      /**
```

```

60      * Array use to store any error found during Validation function
61      * @see Validation()
62      * @var Array
63      */
64     private $validation_error = array();
65
66     /**
67      * Contructor for this method
68      *
69      * <pre>
70      * The constructor will setup the required object for this class
71      * will initiate the database class, the table class and the template
72      * for this class to use
73      *
74      *
75      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
76      * </pre>
77      *
78      * @see db::
79      * @see table
80      * @see template
81      */
82     public function __construct(){
83         $this-> db = new db();
84
85         try {
86             $this-> db_connect = $this-> db-> dbh;
87         } catch (CustomException $e) {
88             $e-> logError();
89         }
90
91         $this-> table = new table();
92         $this-> template = new template();
93     }
94
95     /**
96      * Show will pull a list from the corresponding AdvertTemplate template
97      *
98      * <pre>
99      * This Method will produce a list of all the element corresponding to the result of
AdvertTemplate
100      *
101      * I will only pull rows that are not considered delete
102      * eg. the delete_date field is not "0000-00-00 00:00:00" or set to NULL
103      *
104      * The parameter $filter expects an array with the key being the field to look for and the
105      * value being the the information to filter on
106      *
107      *
108      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
109      * </pre>
110      *
111      * @param String $orderby Which single field is used to oder the output
112      * @param String $direction Which direction os required for the orderby output
113      * @param Array $filter A array of fields to filter, key=$val set (eg array('tile'=>'this
title'))
114      */
115     Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
116
117         $sql = "SELECT template_id,
118                 title,
119                 employmenttype.employmenttype_id,
120                 employmenttype,
121                 template.catagory_id,
122                 category.catagory_name,
123                 office_id,
124                 dept_id,
125                 role_id,
126                 state.name,
127                 storeLoc_id,
128                 start_date,
129                 end_date,
130                 discription,
131                 requirments,
132                 status,
133                 tracking_id,
134                 advertisement_id,
135                 template.create_date,
```

```

136         template.modify_date,
137         template.delete_date
138     FROM template
139     LEFT JOIN category ON template.catagory_id = category.catagory_id
140     LEFT JOIN state ON template.state_id = state.state_id
141     LEFT JOIN employmenttype ON template.employmenttype_id =
employmenttype.employmenttype_id
142     WHERE (template.delete_date = '00-00-0000 00:00:00' OR template.delete_date IS
NULL)" ;
143
144     if(is_array($filter)){
145         foreach($filter AS $key=> $value){
146             if ($value != 'NULL' && !empty($value)){
147                 $sql .= " AND " . $value;
148             }
149         }
150     }
151
152     if($orderby){
153         $sql .= " ORDER BY " . $orderby . " " . $direction;
154     }
155
156     try{
157         $result = $this->db->select($sql);
158     }catch(CustomException $e){
159         echo $e->queryError($sql);
160     }
161
162     return $result;
163 }
164
165 /**
166  * This method will take an array and insert it in the database
167  *
168  * <pre>
169  * This method will insert the formatted information into a database, the format for the array
170  * should be an associated array being the first key should be the table inserting with the keys
171  * for child array the fields that are being inserted too and the values to insert
172  *
173  * Array
174  * (
175  *     [users] => Array
176  *         (
177  *             [name] => Dave
178  *             [surname] => Smith
179  *             [email] => dave@dave.com
180  *         )
181  *     [staff] => Array
182  *         (
183  *             [staff_id] => 1245
184  *             [office_number] => 22
185  *             [drown_code] => bee223
186  *         )
187  * )
188  *
189  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
190  *
191  * </pre>
192  *
193  * @param Array $source
194  *
195  * @return Integer Return last inserted primary id
196  */
197 Private function create($source){
198     try{
199         $this->db_connect->beginTransaction();
200
201         foreach($source['template'] AS $key=> $val){
202             $field[] = $key;
203             $value[] = ":" . $key;
204         }
205
206         $sql = "INSERT INTO template (" . implode(' ', $field) . ") VALUES
(" . implode(' ', $value) . ");" ;
207
208         foreach($source['template'] AS $key=> $val){
209             $exec[":" . $key] = $val;
210         }
211

```

```

212         try{
213             $pid = $this-> db-> insert($sql, $exec);
214         }catch(CustomException $e){
215             throw new CustomException($e-> queryError($sql));
216         }
217         $this-> db_connect-> commit();
218     }
219
220     catch (CustomException $e) {
221         $e-> queryError($sql);
222         $this-> db_connect-> rollBack();
223         return false;
224     }
225
226     return $pid;
227 }
228
229
230 /**
231  * This method will return information as row
232  *
233  * <pre>
234  * This method is you to get a single row of information from the database
235  * based ith the primary id and return it as an array
236  *
237  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
238  * </pre>
239  *
240  * @param Integer $id The primary id of the row to show
241  */
242 Private function read($id){
243
244     $sql = "SELECT template_id,
245 title,
246 employmenttype_id,
247 catagory_id,
248 office_id,
249 dept_id,
250 role_id,
251 state_id,
252 storeLoc_id,
253 start_date,
254 end_date,
255 discription,
256 requirments,
257 status,
258 tracking_id,
259 advertisement_id,
260 create_date,
261 modify_date,
262 delete_date FROM template WHERE template_id = " . $id . " AND (delete_date ='00-00-0000
00:00:00' OR delete_date IS NULL)" ;
263
264
265     $stmt = $this-> db_connect-> prepare($sql);
266     $stmt-> execute();
267
268     try{
269         $result = $this-> db-> select($sql);
270     }catch(CustomException $e){
271         echo $e-> queryError($sql);
272     }
273
274     return $result[0];
275
276 }
277
278
279 /**
280  * This method will take an array and update a row in the database
281  *
282  * <pre>
283  * This method will update the formatted information into the database, the format for the array
284  * should be an associated array being the first key should be the table to be updated with the
keys
285  * for child array the fields that are being updated too and the values to be updated
286  *
287  * Array
288  * (

```

```

289     *      [users] => Array
290     *      (
291     *          [name] => Dave
292     *          [surname] => Smith
293     *          [email] => dave@dave.com
294     *      )
295     *      [staff] => Array
296     *      (
297     *          [staff_id] => 1245
298     *          [office_number] => 22
299     *          [drown_code] => bee223
300     *      )
301     * )
302     *
303     * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
304     *
305     * </pre>
306     *
307     * @param Array $source
308     *
309     * @return Integer Return last inserted primary id
310     */
311 Private function update($source, $id){
312     try{
313         $this-> db_connect-> beginTransaction();
314
315         foreach($source['template'] AS $key=> $val){
316             $field[] = $key." = ":" . $key;
317         }
318
319         $sql = "UPDATE template SET " . implode(', ', $field)." WHERE
template_id =" . $id;
320
321         foreach($source['template'] AS $key=> $val){
322             $exec[":" . $key] = $val;
323         }
324
325         try{
326             $pid = $this-> db-> update($sql, $exec);
327         }catch(CustomException $e){
328             throw new CustomException($e-> queryError($sql));
329         }
330         $this-> db_connect-> commit();
331     }
332
333     catch (CustomException $e) {
334         $e-> queryError($sql);
335         $this-> db_connect-> rollBack();
336         return false;
337     }
338
339     header('Location:advertTemplate.php?action=show&id=' . $id);
340
341 }
342
343 /**
344  * This method will update a row and make the record as deleted
345  *
346  * <pre>
347  * This method will take the id and set the delete_date field to
348  * the current datetime, which will marking it as deleted
349  *
350  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
351  * </pre>
352  *
353  * @param Integer $id The primary id of the row to show
354  * @return Boolean success or failed
355  */
356
357 Private function remove($id){
358     if(empty($id)){
359         return false;
360     }
361
362     $sql = "UPDATE template SET delete_date=NOW() WHERE template_id =" . $id;
363
364     try{
365         $result = $this-> db-> update($sql);

```



```

366         }catch(CustomException $e){
367             echo $e->    queryError($sql);
368         }
369         return true;
370     }
371
372
373     /***** END CRUD METHOD*****/
374
375
376     /**
377      * Show list of information corresponding the to this class
378      *
379      * <pre>This Method will produce a list of all the element corresponding to the result of
380      AdvertTemplate
381      * using the base/table.class.php file, which will format the list into a filtable table that
382      * uses ajax class to change the content on filtering
383      *
384      * There are to response type for this table for the parameter $type
385      *
386      *     TABLE = Will return the content in a table with a filter row and a heading row
387      *     AJAX = Will return just the content after evaluating the filter or heading information
388      *
389      * The parameter $filter expects an array with the key being the field to look for and the
390      * value being the the information to filter on
391      *
392      * <div style="color:red">NOTE: This is generated information from the
393      framework and will need to be corrected if there are any changes</div>
394      * </pre>
395      *
396      * @param String $type Option of type of response for the output of the list
397      * @param String $orderby Which single field is used to oder the output
398      * @param String $direction Which direction os required for the orderby output
399      * @param Array $filter A array of fields to filter, key=TEXT set (eg array('tile'=>'this
400      title'))
401      */
402     public function getAdvertTemplateList($type='TABLE',$orderby=NULL, $direction='ASC',
403     $filter=NULL){
404
405         $result = $this->    lists($orderby, $direction, $filter);
406
407         $this->    table->    removeColumn(array('template_id'));
408
409         switch(strtoupper($type)){
410
411             case 'AJAX' : $this->    table->    setRowsOnly();
412                         $this->    table->    setIdentifier('template_id');
413                         $this->    table->    setIdentifierPage('advertTemplate');
414                         echo $this->    table->    genterateDisplayTable($result);
415
416             BREAK;
417             case 'TABLE' :
418             DEFAULT :
419                 $this->    table->    setHeader(array(
420                     'template_id'=>    'Template Id',
421                     'title'=>    'Title',
422                     'employmenttype_id'=>    'Employmenttype Id',
423                     'catagory_id'=>    'Catagory Id',
424                     'office_id'=>    'Office Id',
425                     'dept_id'=>    'Dept Id',
426                     'role_id'=>    'Role Id',
427                     'state_id'=>    'State Id',
428                     'storeLoc_id'=>    'StoreLoc Id',
429                     'start_date'=>    'Start Date',
430                     'end_date'=>    'End Date',
431                     'discription'=>    'Discription',
432                     'requirments'=>    'Requirments',
433                     'status'=>    'Status',
434                     'tracking_id'=>    'Tracking Id',
435                     'advertisement_id'=>    'Advertisement Id',
436                     'create_date'=>    'Create Date',
437                     'modify_date'=>    'Modify Date',
438                     'delete_date'=>    'Delete Date'));
439
440                 $this->    table->    setFilter(array(
441                     'template_id'=>    'TEXT',
442                     'title'=>    'TEXT',
443                     'employmenttype_id'=>    'TEXT',
444                     'catagory_id'=>    'TEXT',

```

```

442 'office_id'=> 'TEXT',
443 'dept_id'=> 'TEXT',
444 'role_id'=> 'TEXT',
445 'state_id'=> 'TEXT',
446 'storeLoc_id'=> 'TEXT',
447 'start_date'=> 'TEXT',
448 'end_date'=> 'TEXT',
449 'discription'=> 'TEXT',
450 'requirments'=> 'TEXT',
451 'status'=> 'TEXT',
452 'tracking_id'=> 'TEXT',
453 'advertisement_id'=> 'TEXT',
454 'create_date'=> 'TEXT',
455 'modify_date'=> 'TEXT',
456 'delete_date'=> 'TEXT'));
457
458 $this-> table-> setIdentifier('template_id');
459
460 $this-> template-> content(Box($this-> table-
> generateDisplayTable($result),'AdvertTemplate List', 'Shows the current listings for the
AdvertTemplate. To create a new Listing <a href="advertTemplate.php?action=create">Click
Here</a>' ));
461
462 $this-> template-> display();
463
464 }
465
466
467 /**
468  * Show details of a single AdvertTemplate from the template
469  *
470  * <pre>This method will return a template page of the information requested
471  * the method use the template class to format the information ready to display the
472  * the user
473  *
474  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
475  * </pre>
476  * @param Integer $id the primary id of the row to show
477  */
478 Public function showAdvertTemplateDetails($id){
479     $fieldMember = $this-> read($id);
480
481     $this-> template-> page('advertTemplate.tpl.html');
482
483     $this-> templateAdvertTemplateLayout($fieldMember);
484
485     //if($this->checkAdminLevel(1)){
486         $this-> template-> assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='advertTemplate.php?action=edit&id='
487         . $id.\">Edit</
div>\" );
488     //}
489     echo $this-> template-> fetch();
490 }
491
492
493 /**
494  * Show the details ready to edit of a single AdvertTemplate from the template
495  *
496  * <pre>
497  * This method is used to display and editable page to the use, so that they
498  * maybe able to edit any of the fields releated to the row in question.
499  * The method uses the template class to format the information ready to display the
500  * the user
501  *
502  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
503  * </pre>
504  * @param Integer $id The primary id of the row to show
505  */
506 Public function editAdvertTemplateDetails($id){
507
508     $fieldMember = $this-> read($id);
509
510     $name = 'editAdvertTemplate';
511
512     $this-> template-> page('advertTemplate.tpl.html');
513     $this-> template-> assign('FORM-HEADER', '<form
action="advertTemplate.php?action=update&id='
514     . $id.\" method="POST"

```

```

name=" ' . $name. ' ">' );
514
515     $this-> templateAdvertTemplateLayout($fieldMember, true);
516
517     $this-> template-> assign('FUNCTION', " <div class=\"button\"
onclick=\"document. $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='advertTemplate.php?action=show&id= " . $id. "'>Cancel&lt
;/div>" );
518
519     $this-> template-> display();
520 }
521
522 /**
523  * update the information in a single AdvertTemplate from the template
524  *
525  * <pre>
526  * This method is used to take the information from editDepartmentDetails and try to validate
it thought the
527  * validation method and on success will format it ready for input into the database through
the update method
528  *
529  * if the validate fails then the user is show a page that mimics the editDepartmentDetails
method and point out
530  * error in there input
531  *
532  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
533  * </pre>
534  *
535  * @see editDepartmentDetails()
536  * @see Validate()
537  * @see update()
538  * @param Integer $id The primary id of the row to updated
539  */
540 Public function updateAdvertTemplateDetails($id){
541
542     if ($this-> Validate($_REQUEST)){
543
544         $request = $_REQUEST;
545         $table = 'template';
546
547         $save[$table]['title'] = $request['title'];
548         $save[$table]['employmenttype_id'] = $request['employmenttype_id'];
549         $save[$table]['catagory_id'] = $request['catagory_id'];
550         $save[$table]['office_id'] = $request['office_id'];
551         $save[$table]['dept_id'] = $request['dept_id'];
552         $save[$table]['role_id'] = $request['role_id'];
553         $save[$table]['state_id'] = $request['state_id'];
554         $save[$table]['storeLoc_id'] = $request['storeLoc_id'];
555         $save[$table]['start_date'] = formatDateUI($request['start_date']);
556         $save[$table]['end_date'] = formatDateUI($request['end_date']);
557         $save[$table]['discription'] = $request['discription'];
558         $save[$table]['requirments'] = $request['requirments'];
559         $save[$table]['status'] = $request['status'];
560         $save[$table]['tracking_id'] = $request['tracking_id'];
561         $save[$table]['advertisement_id'] = $request['advertisement_id'];
562
563         $save[$table]['modify_date'] = date('Y-m-d h:i:s');
564
565         $this-> update($save, $id );
566         header('Location: advertTemplate.php?action=show&id=' . $id);
567     }else{
568
569         $fieldMember = $this-> valid_field;
570         $error = $this-> validation_error;
571
572         $name = 'editAdvertTemplate';
573
574         $this-> template-> page('advertTemplate.tpl.html');
575
576         foreach($error AS $key=> $value){
577             $this-> template-> assign('err_' . $key, "<span
class=\"error\">" . @implode(' , ' , $error[$key]) . "</span>" );
578         }
579
580         $this-> template-> assign('FORM-HEADER', '<form
action="advertTemplate.php?action=update&id=' . $id. '" method="POST"
name=" ' . $name. ' ">' );
581

```

```

582         $this-> templateAdvertTemplateLayout($fieldMember, true);
583
584         //if($this->admin->checkAdminLevel(1)){
585         $this-> template-> assign('FUNCTION', "      <div
class=\"button\" onclick=\"document.
false\">Update</div><div class=\"button\"
onclick=\"location.href='advertTemplate.php?action=show&id=
; /div>"
);
586         //}
587         $this-> template-> assign('FORM-FOOTER', '</form>' );
588
589         $this-> template-> display();
590     }
591 }
592
593 /**
594  * This method will provide a page to the to add a single row AdvertTemplate to the template
table
595  *
596  * <pre>
597  * The method using the template class to format the information ready to display the
598  * the user, so that they may be able to add any of the fields releated to a row in the
database.
599  * The method uses the template class to format the information ready to display the
600  * the user
601  *
602  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
603  * </pre>
604  */
605 Public function createAdvertTemplateDetails(){
606
607     $name = 'createAdvertTemplate';
608
609     $this-> template-> page('advertTemplate.tpl.html');
610     $this-> template-> assign('FORM-HEADER', '<form
action="advertTemplate.php?action=save" method="POST"
name="' . $name . '">' );
611
612     $this-> templateAdvertTemplateLayout('', true);
613
614     $this-> template-> assign('FUNCTION', "      <div class=\"button\"
onclick=\"document.
class=\"button\"
onclick=\"location.href='advertTemplate.php?action=list'\">Cancel</div>
" );
615
616
617     $this-> template-> display();
618 }
619
620 /**
621  * save the information in a single AdvertTemplate to the template table
622  *
623  * <pre>
624  * This method is used to take the information from createDepartmentDetails and try to validate
it thought the
625  * validation method and on success will format it ready for inserted into the database through
the insert method
626  *
627  * if the validate fails then the user is show a page that mimics the createDepartmentDetails
method and point out
628  * error in there input
629  *
630  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
631  * </pre>
632  *
633  * @see createDepartmentDetails()
634  * @see Validate()
635  * @see update()
636  * @param Integer $id The primary id of the row to updated
637  */
638 Public function saveAdvertTemplateDetails(){
639
640     if ($this-> Validate($_REQUEST)){
641
642         $request = $_REQUEST;
643         $table = 'template';
644
645         $save[$table]['title'] = $request['title'];

```

```

646         $save[$table]['employmenttype_id'] = $request['employmenttype_id'];
647         $save[$table]['catagory_id'] = $request['catagory_id'];
648         $save[$table]['office_id'] = $request['office_id'];
649         $save[$table]['dept_id'] = $request['dept_id'];
650         $save[$table]['role_id'] = $request['role_id'];
651         $save[$table]['state_id'] = $request['state_id'];
652         $save[$table]['storeLoc_id'] = $request['storeLoc_id'];
653         $save[$table]['start_date'] = formatDateUI($request['start_date']);
654         $save[$table]['end_date'] = formatDateUI($request['end_date']);
655         $save[$table]['discription'] = $request['discription'];
656         $save[$table]['requirments'] = $request['requirments'];
657         $save[$table]['status'] = $request['status'];
658         $save[$table]['tracking_id'] = $request['tracking_id'];
659         $save[$table]['advertisement_id'] = $request['advertisement_id'];
660
661         $save[$table]['create_date'] = date('Y-m-d h:i:s');
662
663         $id = $this-> create($save);
664         header('Location: advertTemplate.php?action=show&id=' . $id);
665     }else{
666
667         $fieldMember = $this-> valid_field;
668
669         $error = $this-> validation_error;
670
671         $name = 'createAdvertTemplate';
672
673         $this-> template-> page('advertTemplate.tpl.html');
674
675         foreach($error AS $key=> $value){
676             $this-> template-> assign('err_'. $key, "<span
677 class=\"error\">" . @implode(' ', $value) . "</span>");
678         }
679
680         $this-> template-> assign('FORM-HEADER', '<form
681 action="advertTemplate.php?action=save" method="POST"
682 name="' . $name . '">');
683
684         $this-> templateAdvertTemplateLayout($fieldMember, true);
685
686         //if($this->admin->checkAdminLevel(1)){
687             $this-> template-> assign('FUNCTION', " <div
688 class=\"button\" onclick=\"document. $name.submit(); return
689 false\">Update</div><div class=\"button\"
690 onclick=\"location.href='advertTemplate.php'>Cancel</div>
691 " );
692         }
693
694         $this-> template-> assign('FORM-FOOTER', '</form>');
695
696         $this-> template-> display();
697     }
698 }
699
700 /**
701  * Set a row to be marked as deleted
702  *
703  * <pre>
704  * This method will take the id and set the delete_date field to
705  * the current datetime, which will marking it as deleted
706  *
707  * <div style="color:red">NOTE: This is generated information from the
708  * framework and will need to be corrected if there are any changes</div>
709  * </pre>
710  * @param Integer $id The primary id of the row to marked as delete
711  */
712 Public function deleteAdvertTemplateDetails($id){
713     $this-> remove($id);
714     header('Location: advertTemplate.php');
715 }
716
717 /**
718  * This method assigns the associate array values to the template
719  *
720  * <pre>
721  * This method is used to incorporate the standards elements of the templates to a single
722  * function across all tempatled methods
723  * <div style="color:red">NOTE: This is generated information from the
724  * framework and will need to be corrected if there are any changes</div>
725  * </pre>
726  * @todo find out what $inputArray is used for
727  */

```

```

718      * @param Array $fielddata An associative array of fields that need to be assigned to the
719      template object
720      * @param Boolean $input If false then just assign the value if true the add the value to
721      corresponding form element
722      * @param Array $inputArray Not sure :S
723      */
724      private function templateAdvertTemplateLayout($fieldMember, $input = false, $inputArray=array())
725      {
726          $id = @$fieldMember['template_id'];
727
728          @$this-> template-> assign('title', ($input)? $this-> template-
729          > input('text', 'title', $fieldMember['title']):$fieldMember['title']);
730          @$this-> template-> assign('employmenttype_id', ($input)? $this-
731          > getSelectListOfEmploymentType($fieldMember['employmenttype_id'], True):$this-
732          > getSelectListOfEmploymentType($fieldMember['employmenttype_id']));
733          @$this-> template-> assign('catagory_id', ($input)? $this-
734          > getSelectListOfCategory($fieldMember['catagory_id'], True):$this-
735          > getSelectListOfCategory($fieldMember['catagory_id']));
736          @$this-> template-> assign('office_id', ($input)? $this-> template-
737          > input('text', 'office_id', $fieldMember['office_id']):$fieldMember['office_id']);
738          @$this-> template-> assign('dept_id', ($input)? $this-> template-
739          > input('text', 'dept_id', $fieldMember['dept_id']):$fieldMember['dept_id']);
740          @$this-> template-> assign('role_id', ($input)? $this-> template-
741          > input('text', 'role_id', $fieldMember['role_id']):$fieldMember['role_id']);
742          @$this-> template-> assign('state_id', ($input)? $this-
743          > getSelectListOfStates($fieldMember['state_id'], True):$this-
744          > getSelectListOfStates($fieldMember['state_id']));
745          @$this-> template-> assign('storeLoc_id', ($input)? $this-> template-
746          > input('text', 'storeLoc_id', $fieldMember['storeLoc_id']):$fieldMember['storeLoc_id']);
747          @$this-> template-> assign('start_date', ($input)? $this-> template-
748          > input('text', 'start_date', $fieldMember['start_date']):$fieldMember['start_date']);
749          @$this-> template-> assign('end_date', ($input)? $this-> template-
750          > input('text', 'end_date', $fieldMember['end_date']):$fieldMember['end_date']);
751          @$this-> template-> assign('discription', ($input)? $this-> template-
752          > input('text', 'discription', $fieldMember['discription']):$fieldMember['discription']);
753          @$this-> template-> assign('requirments', ($input)? $this-> template-
754          > input('text', 'requirments', $fieldMember['requirments']):$fieldMember['requirments']);
755          @$this-> template-> assign('status', ($input)? $this-> template-
756          > input('text', 'status', $fieldMember['status']):$fieldMember['status']);
757          @$this-> template-> assign('tracking_id', ($input)? $this-> template-
758          > input('text', 'tracking_id', $fieldMember['tracking_id']):$fieldMember['tracking_id']);
759          @$this-> template-> assign('advertisement_id', ($input)? $this-> template-
760          > input('text', 'advertisement_id', $fieldMember['advertisement_id']):$fieldMember['advertisement_id']);
761          @$this-> template-> assign('create_date', ($input)? $this-> template-
762          > input('text', 'create_date', $fieldMember['create_date']):$fieldMember['create_date']);
763          @$this-> template-> assign('modify_date', ($input)? $this-> template-
764          > input('text', 'modify_date', $fieldMember['modify_date']):$fieldMember['modify_date']);
765          @$this-> template-> assign('delete_date', ($input)? $this-> template-
766          > input('text', 'delete_date', $fieldMember['delete_date']):$fieldMember['delete_date']);
767
768          /*if(isset($id)){
769              $this->template->assign('COMMENTS', $this->comment-
770              >getCommentBox($id, 'advertTemplate'));
771          }*/
772      }
773
774      /**
775      * This medthod is used to validate inputs from form information
776      * <pre>
777      * This method will first check the if the fields are in the valid_field array and strip out
778      any that are not
779      * Then it check that fields that require a value in them from the fields_required have a
780      value, if not add an error to validation_error array
781      * Then it will check all the values to find out if the value match the type found in the
782      fields_validation_type array, if not add an error to validation_error array
783      * <div style="color:red">NOTE: This is generated information from the
784      framework and will need to be corrected if there are any changes</div>
785      * </pre>
786      * @see fields
787      * @see fields_required
788      * @see fields_validation_type
789      * @see validation_error
790      * @param Array $request
791      */

```

```

769 public function Validate($request){
770
771     unset($this-> valid_field);
772     unset($this-> validation_error);
773     $isvalid = True;
774
775     $validfields = $this-> fields;
776     $requiredfields = $this-> fields_required;
777     $fieldsvalidationtype = $this-> fields_validation_type;
778
779     foreach ($request AS $key=> $value){ //lets strip put unwanted or security violation
fields
780         if(in_array($key, $validfields)){
781             $this-> valid_field[$key] = $value; //pure fields
782         }
783     }
784
785     foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxes
786         if(!isset($this-> valid_field[$value])){
787             $this-> valid_field[$value] = '';
788         }
789     }
790
791     if(count($requiredfields) > 0 ){
792         foreach($requiredfields AS $value){ // lets check all the required fields have a value
793             if (empty($this-> valid_field[$value]) || $this-> valid_field[$value] ==
'NULL'){
794                 $this-> validation_error[$value][] = 'Field is Required'; //error field
795                 $isvalid = false;
796             }
797         }
798     }
799
800
801
802     //now lets validate
803     foreach ($this-> valid_field AS $key=> $value){
804         $value = trim($value);
805         if(!empty($value)){ // don't check if empty, already done in required check
806
807             switch(@$fieldsvalidationtype[$key]){
808                 case 'TEXTAREA': if (strlen($value) > 1024) {
809                     $this-> validation_error[$key][] = 'Field longer than 1024
characters';
810                     $isvalid = false;
811                 } break;
812                 case 'TEXT': if (strlen($value) > 1024) {
813                     $this-> validation_error[$key][] = 'Field longer than 1024
characters';
814                     $isvalid = false;
815                 } break;
816                 case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
817                     $this-> validation_error[$key][] = 'not a valid SAP number';
818                     $isvalid = false;
819                 } break;
820                 case 'DECIMAL': if (!is_numeric($value)) {
821                     $this-> validation_error[$key][] = 'Decimal value expected';
822                     $isvalid = false;
823                 } break;
824                 case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"
) && ($value != 1)) {
825                     $this-> validation_error[$key][] = 'Please check';
826                     $isvalid = false;
827                 } break;
828                 case 'INT': if (!is_numeric($value) && $value != 'NULL' ){
829                     $this-> validation_error[$key][] = 'Numeric value expected';
830                     $isvalid = false;
831                 } break;
832                 case 'DATE': //$date = str_replace('/', '-', $value);
833                     //$date = str_replace("\\", '-', $date);
834                     @list($day, $month, $year) = explode('/', $value);
835                     if(!checkdate($month,$day, $year)){
836                         $this-> validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
837                         $isvalid = false;
838                     } break;
839                 case 'YEAR': if(!checkYear($value)){
840                     $this-> validation_error[$key][] = 'incorrect year
format, expecting yyyy';

```



```

841         $isvalid = false;
842     } break;
843 }
844     }
845 }
846 }
847
848     return $isvalid;
849 }
850
851 /**
852  * This Method will either the category_name feild or a select box of category_name fields
853  * This Methos is based in the category_id of the category table, it will return either the
854  information
855  * in the category_name field as a string or a select box with the id selected
856  * @param Integer $id the id of the row we are looking for
857  * @param Boolean $selectBox True will return it as a select box with field selected, else just
858  the field
859  * @return string
860  */
861 public function getSelectListOfCategory($id, $selectBox=NULL){
862
863     if($selectBox == false){
864         return $this-> template-> getListTable('category', $id, 'catagory_id',
865 'category_name');
866     }else{
867         $select = "<select name=\"category_id\">" ;
868         $select .= "<option value=\"NULL\"></option>" ;
869         $select .= $this-> template-> getListTable('category', $id, 'catagory_id',
870 'category_name', $selectBox);
871         $select .= "</select>" ;
872         return $select;
873     }
874 }
875
876 /**
877  * This Method will either the name feild or a select box of state fields
878  * This Methos is based in the state_id of the state table, it will reurn either the information
879  * in the state name field as a string or a select box with the id selected
880  * @param Integer $id the id of the row we are looking for
881  * @param Boolean $selectBox True will return it as a select box with field selected, else just
882  the field
883  * @return string
884  */
885 public function getSelectListOfStates($id, $selectBox=NULL){
886
887     if($selectBox == false){
888         return $this-> template-> getListTable('state', $id, 'state_id', 'name');
889     }else{
890         $select = "<select name=\"state_id\">" ;
891         $select .= "<option value=\"NULL\"></option>" ;
892         $select .= $this-> template-> getListTable('state', $id, 'state_id', 'name',
893 $selectBox);
894         $select .= "</select>" ;
895         return $select;
896     }
897 }
898
899 /**
900  * This Method will either the employmenttype feild or a select box of employmenttype fields
901  * This Methos is based in the employmenttype_id of the employmenttype table, it will reurn
902  either the information
903  * in the employmenttype field as a string or a select box with the id selected
904  * @param Integer $id the id of the row we are looking for
905  * @param Boolean $selectBox True will return it as a select box with field selected, else just
906  the field
907  * @return string
908  */
909 public function getSelectListOfEmploymentType($id, $selectBox=NULL){
910
911     if($selectBox == false){
912         return $this-> template-> getListTable('employmenttype', $id,
913 'employmenttype_id', 'employmenttype');
914     }else{
915         $select = "<select name=\"employmenttype_id\">" ;
916         $select .= "<option value=\"NULL\"></option>" ;

```



```
912         $select .= $this->template->getListTable('employmenttype', $id,  
'employmenttype_id', 'employmenttype', $selectBox);  
913         $select .= "</select>" ;  
914         return $select;  
915     }  
916 }  
917 }
```

# File Source for errorhandler.class.php

Documentation for this file is available at [errorhandler.class.php](http://www.phpdoc.org/package/PhpDocumentor/errorhandler.class.php)

```
1  <?php
2  /**
3   * CustomException Class,
4   * <br />
5   * Generates a custom exception<br />
6   *
7   * Example:<br />
8   * <br />
9   * throw an exception <br />
10  * <br />
11  * if(empty($value)){<br />
12  *     throw new CustomException('value can not be empty');<br />
13  * }<br />
14  *
15  * try/catch<br />
16  * <br />
17  * try{<br />
18  *     $db->query('SELECT * FROM notable ');<br />
19  * }catch(CustomException $e){<br />
20  *     echo $e->logError();<br />
21  * }<br />
22  *
23  * @author Jason Stewart <jason@lexxcom.com.au>
24  * @version 1.0
25  * @package PeopleScope
26  */
27
28  /**
29   * Generates a custom exception
30   *
31   * @package PeopleScope
32   * @subpackage Base
33   */
34  class CustomException extends Exception{
35
36      public function messageError(){
37          $str = $this->    getMessage(). " File: "           . $this->    getFile(). " line: "
38          . $this->    getLine();
39          return $str;
40      }
41
42      /**
43       * Generate a formatted excption error
44       *
45       * @return string Html Format
46       */
47      public function logError(){
48
49          $str = "<div style=\"font-family: Arial, Helvetica, sans-serif; font-size:
50 15px; padding:10px; color:#000; background-color: #ffcccc; width:100%; border:solid 1px
51 #000\"><h4>CustomException Error Information</h4>"
52          . "<div> Message :: "           . $this->    getMessage(). "</div>"
53          . "<div>File :: "           . $this->    getFile(). "</div>"
54          . "<div>Line :: "           . $this->    getLine(). "</div></pre>"
55          . "</div>" ;
56
57          if(DEBUG){
58              $trace = "<pre>"           . print_r($this-
59              > getTrace(),1). "</pre>" ;
60              $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
61 size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
62 #000\"><h4>Error Back Trace</h4><pre>"
63              . htmlentities (print_r($trace,1), ENT_COMPAT) .
64              . "</pre></div>" ;
65          }
66
67          return $str;
68      }
69  }
```

```

62 }
63 }
64 /**
65  * Will append the query being used to the begining of a logError output
66  *
67  * This is used to create an exception error for query execution, to produce
68  * error that also have the query displayed with in the error output
69  *
70  * @param $query Sql query that was being used at the time of execution
71  */
72 public function queryError($query){
73     $lines = explode("\n", $query);
74     $count = 1;
75     $lineQuery = '';
76     foreach ($lines AS $line){
77         $lineQuery .= str_pad($count++, 3)."\t" . $line."\n" ;
78     }
79     echo "<div style=\"font-family: Arial, Helvetica, sans-serif; font-size: 15px;
color:#000; padding:10px; background-color: #eccc00; width:100%; border:solid 1px
#000\"><h2>SQL Statement</h2><pre>" . $lineQuery."</div>" ;
80     echo $this-> logError();
81 }
82 }
83
84 set_error_handler("exception_error_handler" );
85 function exception_error_handler($errno, $errstr, $errfile, $errline ) {
86     throw new CustomException($errstr, 0, $errno, $errfile, $errline);
87 }
88
89 set_error_handler("myErrorHandler" , E_ALL);
90
91 function exception_handler($exception) {
92     echo "Uncaught exception: " , $exception-> getMessage(), "\n" ;
93 }
94
95 set_exception_handler('exception_handler');
96
97 function myErrorHandler($errno, $errstr, $errfile, $errline)
98 {
99     if (!(error_reporting() & $errno)) {
100         // This error code is not included in error_reporting
101         return;
102     }
103
104     switch ($errno) {
105         case E_ERROR:
106         case E_CORE_ERROR :
107         case E_USER_ERROR:
108             $str = "<div style=\"border: green 1px solid; font-family: Arial,
Helvetica, sans-serif; font-size: 15px; color:#000; padding:10px; background-color: #8AE234; width:100%;
border:solid 3px red\"><h4>PHP ERROR:: Error
Information[ " . $errno."]</h4>"
109                 "<div> Message ::
<strong>" . $errstr."</strong></div><br />
110                 <div>File :: " . $errfile."</div>
111                 <div>Line :: " . $errline."</div></pre>
112                 </div>" ;
113                 if(DEBUG){
114                     $trace =
115                     "<pre>" . print_r(debug_backtrace(),1)."</pre>" ;
116                     $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
#000\"><h4>Error Back Trace</h4><pre>" .
117                     htmlentities (print_r($trace,1), ENT_COMPAT) .
118                     "</pre></div>" ;
119                 }
120                 return ($str);
121                 break;
122         case E_WARNING:
123         case E_CORE_WARNING:
124         case E_USER_WARNING:
125             $str = "<div style=\"border: green 3px solid; font-family: Arial,
Helvetica, sans-serif; font-size: 15px; color:#000; padding:10px; background-color: #8AE234; width:100%;
border:solid 3px green\"><h4>PHP WARNING:: Error
Information[ " . $errno."]</h4>"
126                 "<div> Message ::
<strong>" . $errstr."</strong></div><br />
127                 <div>File :: " . $errfile."</div>
128                 <div>Line :: " . $errline."</div></pre>
129                 </div>" ;

```

```

129         if(DEBUG){
130             $trace =
131             "<pre>"
132             .print_r(debug_backtrace(),1)."</pre>"
133             $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
134             size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
135             #000\"><h4>Error Back Trace</h4><pre>"
136             htmlentities (print_r($trace,1), ENT_COMPAT) .
137             "</pre></div>"
138         }
139         break;
140     case E_NOTICE:
141     case E_CORE_NOTICE:
142     case E_USER_NOTICE:
143         $str = "<div style=\"border: green 1px solid; font-family: Arial,
144         Helvetica, sans-serif; font-size: 15px; color:#000; padding:10px; background-color: #8AE234; width:100%;
145         border:solid 3px green\"><h4>PHP NOTICE:: Error
146         Information[ "
147         . $errno." ]</h4>"
148         "<div> Message ::
149         <strong>"
150         . $errstr."</strong></div><br />
151         <div>File :: "
152         . $errfile."</div>
153         <div>Line :: "
154         . $errline."</div></pre>
155         </div>"
156         ;
157         if(DEBUG){
158             $trace =
159             "<pre>"
160             .print_r(debug_backtrace(),1)."</pre>"
161             $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
162             size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
163             #000\"><h4>Error Back Trace</h4><pre>"
164             htmlentities (print_r($trace,1), ENT_COMPAT) .
165             "</pre></div>"
166         }
167         break;
168     default:
169         $str = "<div style=\"border: green 1px solid; font-family: Arial,
170         Helvetica, sans-serif; font-size: 15px; color:#000; padding:10px; background-color: #8AE234; width:100%;
171         border:solid 3px green\"><h4>PHP OTHER:: Error
172         Information[ "
173         . $errno." ]</h4>"
174         "<div> Message ::
175         <strong>"
176         . $errstr."</strong></div><br />
177         <div>File :: "
178         . $errfile."</div>
179         <div>Line :: "
180         . $errline."</div></pre>
181         </div>"
182         ;
183         if(DEBUG){
184             $trace =
185             "<pre>"
186             .print_r(debug_backtrace(),1)."</pre>"
187             $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
188             size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
189             #000\"><h4>Error Back Trace</h4><pre>"
190             htmlentities (print_r($trace,1), ENT_COMPAT) .
191             "</pre></div>"
192         }
193         break;
194     }
195     echo $str;
196     /* Don't execute PHP internal error handler */
197     return true;
198 }

```

# File Source for template.old.class.php

Documentation for this file is available at [template.old.class.php](http://www.phpdoc.org/package/PhpDocumentor)

```
1  <?php
2  /**
3   * Template Class,
4   * <br />
5   * This class is used to take a input to generate templates<br />
6   *
7   * Example:<br />
8   * <br />
9   * $template = new template('template/index.html');<br />
10  * <br />
11  * $template->assign('var1', 'Employment list');<br />
12  * $template->assignArray(array{'var2'=>'John', 'var3'=>'mary',
13  * 'var4'=>'<strong>frank</strong>'});<br />
14  * <br />
15  * $ArrayVars[0]['name'] = 'john';<br />
16  * $ArrayVars[0]['age'] = '14';<br />
17  * $ArrayVars[1]['name'] = 'mary';<br />
18  * $ArrayVars[1]['age'] = '42';<br />
19  * $ArrayVars[2]['name'] = 'frank';<br />
20  * $ArrayVars[2]['age'] = '98';<br />
21  * <br />
22  * $template->assignRepeat('agelist', $ArrayVars);<br />
23  * $template->replace('..', '...');<br />
24  * <br />
25  * $template->display();<br />
26  * <br />
27  * @author Jason Stewart <jason@lexxcom.com.au>
28  * @version 1.0
29  * @package PeopleScope
30  */
31
32 /**
33  * required error handler
34  */
35 require_once 'errorhandler.class.php';
36
37 /**
38  * This class is used to take a input to generate templates
39  *
40  * @package PeopleScope
41  * @subpackage Base
42  */
43 class template {
44     /**
45      * Location of template file
46      * @var String
47      */
48     var $template_file;
49
50     /**
51      * array of assigned values to a template
52      * @var Array
53      */
54     var $assigned = array();
55
56     /**
57      * Array of values that should be replaced in template
58      * @var Array
59      */
60     var $replacer = array();
61
62     /**
63      * Array of Repeat region
64      * @todo should be removed as we are not using Dreamweaver template anymore
65      * @var Array
66      */
67 }
```

```

67     var $repeatRegion=array();
68
69     /**
70      * Constructor for template class
71      *
72      * @param String $file Path to current template file
73      * @return true
74      */
75     public function __construct($file=null){
76         $this-> template($file);
77         return true;
78     }
79
80     /**
81      * Constructor php4 for template class
82      *
83      * @param String $file Path to current template file
84      * @return true
85      */
86     public function template($file=null){
87         if($file){
88             $this-> set($file);
89         }
90         return true;
91     }
92
93     /**
94      * Set the current template path to a new template file
95      *
96      * @param String $file new path to template
97      * @return true
98      */
99     public function set($file){
100
101         if(!is_file($file)){
102             throw new CustomException($file.': file not found');
103         }
104         $filetmp = file($file);
105         $fullfile = '';
106         foreach($filetmp as $value){
107             $fullfile .= $value."\n" ;
108         }
109         $this-> template_file = $file;
110         return true;
111
112         define(DEBUG, true);
113     }
114
115     /**
116      * Used to assign a value element to a generated template
117      *
118      * @param String $name name of element
119      * @param String $content content to be replced with
120      * @return true
121      */
122     public function assign($name, $content){
123         $this-> assigned[$name] = $content;
124         return true;
125     }
126
127     /**
128      * Assign Blank is used if not using a template, example such as XML output
129      *
130      * example:<br />
131      *
132      * $template = new template();<br />
133      * $template2->assignBlank('This is a test');<br />
134      *
135      * @param String $content element to be displayed
136      * @return true;
137      */
138     public function assignBlank($content){
139         $this-> assigned['blank'] = $content;
140         return true;
141     }
142
143     /**
144      * Will assign to a repeating element from an assigned array <br />
145      * based on array row and element name eg.
146      * <br />

```

```

147 * <code>
148 * array{
149 *     [0]=> array{
150 *         [name]=>'john'
151 *         [age]=>'14'
152 *     }
153 *     [1]=> array{
154 *         [name]=>'simon'
155 *         [age]=>'23'
156 *     }
157 * }
158 * </code>
159 *
160 * @param String $name Repeat assignment name
161 * @param Array $content Array of arrays values
162 * @return true
163 */
164 public function assignRepeat($name, $content){
165     $this-> repeatRegion[$name] = $content;
166     return true;
167 }
168
169 /**
170 * An Array of elements to be added to the template
171 * array{'jason'=>'john', 'age'=>'14'}
172 *
173 * @param Array $assignedArray array of element to be displayed
174 * @return true
175 */
176 public function assignArray($assignedArray){
177     $this-> assigned = $assignedArray;
178     return true;
179 }
180
181 /**
182 * Will replace the string across the entire template
183 *
184 * replace happens before compiling, so it is possible to change a tag on the fly
185 *
186 * @param String $string Sting to find
187 * @param String $value Value to be replaced with
188 * @return true
189 */
190 public function replace($string, $value='') {
191     $this-> replacer[] = array('string'=> $string, 'value'=> $value);
192     return true;
193 }
194
195 /**
196 * Build template with assigned values
197 *
198 * @param $content Content of the template
199 *
200 * @return Sting HTML Generated from gathered information
201 */
202 private function BuildAssigned($content) {
203
204     // default add jquery to the head of the file
205     preg_match_all( '/<head>(.*?)</head>/siU' , $content, $head);
206
207     $newheader = '<head><script type="text/javascript"
src="/js/jquery.js"></script>' . "\n" . $head[1][0]. "</head>" ;
208     $content = preg_replace( '/<head>(.*?)</head>/siU' , $newheader, $content);
209
210     // default add debugging tool to the body of the file
211     preg_match_all( '/<body>(.*?)>/siU' , $content, $body);
212     $debug = $body[0][0]. showVars();
213     $content = preg_replace( '/<body>(.*?)>/siU' , $debug, $content);
214
215     if( preg_match( '/{\{.*repeat\=(.*?)\*\}/' , $content ) ){
216
217         preg_match_all( '/{\{.*repeat\=(.*?)\*\}/' , $content, $variable);
218
219         foreach($variable[1] AS $value){
220             $compiledtemplate = '';
221             // lets get our template
222             preg_match_all( '/{\{.*repeat\=' . $value . '\*\}(.*?)\{\{.*repeat\*\}/siU' ,
$content, $repeatContent);
223             $rtemplate = $repeatContent[1][0];
224

```

```

225         // does the section of template have an assignment var
226         if( preg_match( '/\{*(.*?)\*\/', $rtemplate) ){
227             preg_match_all( '/\{*(.*?)\*\/', $rtemplate, $vars); //get list of vars
228             //get the var set for this section of template
229             foreach($this-> repeatRegion[trim($value)] AS $rs){
230                 $rt = $rtemplate;
231                 //Merge vars to with content
232                 foreach($vars[1] AS $key=> $assignname){
233                     $rt =str_replace($vars[0][$key], $rs[$assignname], $rt );
234                 }
235                 $compiledtemplate .= $rt."<br />"
236             }
237         }
238         $content = preg_replace( '/\{*(repeat\='.$value.'\*\})(.*)\{*(repeat\*\/)/siU'
239 , $compiledtemplate, $content);
240     }
241 }
242
243 if( preg_match( '/\{*(.*?)\*\/', $content ) ){
244     preg_match_all( '/\{*(.*?)\*\/', $content, $variable);
245
246     foreach($variable[0] AS $key=> $val){
247         if (isset($this-> assigned[trim($variable[1][$key])]){
248             $content =str_replace($variable[0][$key], $this-
249 > assigned[trim($variable[1][$key])], $content );
250         }
251     }
252 }
253
254 if( preg_match( '/\{?(.*?)\?\/', $content ) ){
255     preg_match_all( '/\{?(.*?)\?\/', $content, $code);
256
257     foreach($code[0] AS $key=> $val){
258         ob_start();
259         eval($code[1][$key]);
260         $result = ob_get_contents();
261         ob_clean();
262         $content =str_replace($code[0][$key], $result, $content );
263     }
264 }
265
266 return $content;
267 }
268
269 /**
270  * Will return the full generated page template as a variable
271  *
272  * @return Sting html Generated from gathered information
273  */
274 public function fetch(){
275     if($this-> template_file){
276         if(!$template = file_get_contents($this-> template_file)){
277             throw new CustomException('$this->template_file: Unable to read file contents' );
278         }
279
280         foreach($this-> replacer AS $key=> $value){
281             $template = str_replace($value['string'], $value['value'], $template);
282         }
283
284         $template=$this-> BuildAssigned($template);
285
286         if(DEBUG){
287             $body = parseArray($template, '<body' , ">" );
288             $template = str_replace($body, showVars(), $template);
289         }
290     }else{
291         if(!DEBUG){
292             $vars = showVars();
293             $template = '<html><body>' . $vars.$this-
294 > assigned['blank']. '</body></html>' ;
295         }else{
296             $template = $this-> assigned['blank'];
297         }
298     }
299     return $template;
300 }
301

```



```
302
303      /**
304       * Will print out the full generated page template
305       *
306       */
307      public function display(){
308          echo $this-> fetch();
309      }
310
311  }
312  ?>
```

# File Source for category.class.php

Documentation for this file is available at [category.class.php](http://category.class.php)

```
1  <?php
2  /**
3   * Category Class
4   * <pre>
5   * This class is based on the table category
6   *
7   * <div style="color:red">NOTE: This is generated information from the framework
and will need to be corrected if there are any changes</div>
8   * </pre>
9   *
10  * @author Jennifer Erator <jason@lexxcom.com>
11  * @version 1.1 of the Framework generator
12  * @package PeopleScope
13  */
14
15  class category {
16
17      /**
18       * Connect to PDO object through database class
19       * @var Object
20       */
21      private $db_connect;
22
23      /**
24       * Database class object
25       * @var Object
26       */
27      private $db;
28
29      /**
30       * Table class object
31       * @var Object
32       */
33      public $table;
34
35      /**
36       * Template class object
37       * @var Object
38       */
39      public $template;
40
41      /**
42       * Array of field used in the database if not in this list is dropped from insert or update
43       * @var Array
44       */
45      private $fields =array('catagory_id', 'catagory_name', 'create_date', 'modify_date',
'delete_date');
46
47      /**
48       * Array of feilds require information when validating
49       * @var Array|null
50       */
51      private $fields_required = array('catagory_name');
52
53      /**
54       * Array of feilds and there types that are check when validating
55       * @var Array|null
56       */
57      private $fields_validation_type = array ('catagory_id'=> 'INT', 'catagory_name'=> 'TEXT',
'create_date'=> 'TEXT', 'modify_date'=> 'TEXT', 'delete_date'=> 'TEXT');
58
59      /**
60       * Array use to store any error found during Validation function
61       * @see Validation()
62       * @var Array
63       */
64      private $validation_error = array();
```

```

65
66 /**
67  * Constructor for this method
68  *
69  * <pre>
70  * The constructor will setup the required object for this class
71  * will initiate the database class, the table class and the template
72  * for this class to use
73  *
74  *
75  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
76  * </pre>
77  *
78  * @see db::
79  * @see table
80  * @see template
81  */
82 public function __construct(){
83     $this-> db = new db();
84
85     try {
86         $this-> db_connect = $this-> db-> dbh;
87     } catch (CustomException $e) {
88         $e-> logError();
89     }
90
91     $this-> table = new table();
92     $this-> template = new template();
93
94 }
95
96 /**
97  * Show will pull a list from the corresponding Category category
98  *
99  * <pre>
100  * This Method will produce a list of all the element corresponding to the result of Category
101  *
102  * I will only pull rows that are not considered delete
103  * eg. the delete_date field is not "0000-00-00 00:00:00" or set to NULL
104  *
105  * The parameter $filter expects an array with the key being the field to look for and the
106  * value being the the information to filter on
107  *
108  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
109  * </pre>
110  *
111  * @param String $orderby Which single field is used to oder the output
112  * @param String $direction Which direction os required for the orderby output
113  * @param Array $filter A array of fields to filter, key=$val set (eg array('tile'=>'this
title'))
114  */
115 Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
116
117     $sql = "SELECT catagory_id,
118 catagory_name,
119 create_date,
120 modify_date,
121 delete_date FROM category WHERE (delete_date ='00-00-0000 00:00:00' OR delete_date IS NULL)" ;
122
123     if(is_array($filter)){
124         foreach($filter AS $key=> $value){
125             if ($value != 'NULL' && !empty($value)){
126                 $sql .= " AND " . $value;
127             }
128         }
129     }
130
131     if($orderby){
132         $sql .= " ORDER BY " . $orderby . " . $direction;
133     }
134
135     try{
136         $result = $this-> db-> select($sql);
137     }catch(CustomException $e){
138         echo $e-> queryError($sql);
139     }
140
141     return $result;

```

```

142     }
143
144     /**
145     * This method will take an array and insert it in the database
146     *
147     * <pre>
148     * This method will insert the formatted information into a database, the format for the array
149     * should be an associated array being the first key should be the table inserting with the keys
150     * for child array the fields that are being inserted too and the values to insert
151     *
152     * Array
153     * (
154     *     [users] => Array
155     *     (
156     *         [name] => Dave
157     *         [surname] => Smith
158     *         [email] => dave@dave.com
159     *     )
160     *     [staff] => Array
161     *     (
162     *         [staff_id] => 1245
163     *         [office_number] => 22
164     *         [drown_code] => bee223
165     *     )
166     * )
167     *
168     * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
169     *
170     * </pre>
171     *
172     * @param Array $source
173     *
174     * @return Integer Return last inserted primary id
175     */
176     Private function create($source){
177         try{
178             $this-> db_connect-> beginTransaction();
179
180             foreach($source['category'] AS $key=> $val){
181                 $field[] = $key;
182                 $value[] = ":" . $key;
183             }
184
185             $sql = "INSERT INTO category (" . implode(' ', $field) . ") VALUES
(" . implode(' ', $value) . ");" ;
186
187             foreach($source['category'] AS $key=> $val){
188                 $exec[":" . $key] = $val;
189             }
190
191             try{
192                 $spid = $this-> db-> insert($sql, $exec);
193             }catch(CustomException $e){
194                 throw new CustomException($e-> queryError($sql));
195             }
196             $this-> db_connect-> commit();
197         }
198
199         catch (CustomException $e) {
200             $e-> queryError($sql);
201             $this-> db_connect-> rollBack();
202             return false;
203         }
204
205         return $spid;
206     }
207
208
209     /**
210     * This method will return information as row
211     *
212     * <pre>
213     * This method is you to get a single row of information from the database
214     * based ith the primary id and return it as an array
215     *
216     * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
217     * </pre>
218     *

```

```

219     * @param Integer $id The primary id of the row to show
220     */
221     Private function read($id){
222
223         $sql = "SELECT catagory_id,
224 catagory_name,
225 create_date,
226 modify_date,
227 delete_date FROM category WHERE catagory_id = " . $id . " AND (delete_date ='00-00-0000
00:00:00' OR delete_date IS NULL)" ;
228
229
230         $stmt = $this-> db_connect-> prepare($sql);
231         $stmt-> execute();
232
233         try{
234             $result = $this-> db-> select($sql);
235         }catch(CustomException $e){
236             echo $e-> queryError($sql);
237         }
238
239         return $result[0];
240
241     }
242
243
244     /**
245     * This method will take an array and update a row in the database
246     *
247     * <pre>
248     * This method will update the formated information into the database, the format for the array
249     * should be an associated array being the first key should be the table to be updated with the
250     keys
251     * for child array the fields that are being updated too and the values to be updated
252     *
253     * Array
254     * (
255     *     [users] => Array
256     *         (
257     *             [name] => Dave
258     *             [surname] => Smith
259     *             [email] => dave@dave.com
260     *         )
261     *     [staff] => Array
262     *         (
263     *             [staff_id] => 1245
264     *             [office_number] => 22
265     *             [drown_code] => bee223
266     *         )
267     *     )
268     * <div style="color:red">NOTE: This is generated information from the
269     framework and will need to be corrected if there are any changes</div>
270     * </pre>
271     *
272     * @param Array $source
273     *
274     * @return Integer Return last inserted primary id
275     */
276     Private function update($source, $id){
277         try{
278             $this-> db_connect-> beginTransaction();
279
280             foreach($source['category'] AS $key=> $val){
281                 $field[] = $key." = ":" . $val;
282             }
283
284             $sql = "UPDATE category SET " . implode(', ', $field)." WHERE
catagory_id =" . $id;
285
286             foreach($source['category'] AS $key=> $val){
287                 $exec[":" . $key] = $val;
288             }
289
290             try{
291                 $pid = $this-> db-> update($sql, $exec);
292             }catch(CustomException $e){
293                 throw new CustomException($e-> queryError($sql));
294             }

```

```

295         $this-> db_connect-> commit();
296     }
297
298     catch (CustomException $e) {
299         $e-> queryError($sql);
300         $this-> db_connect-> rollBack();
301         return false;
302     }
303
304     header('Location:category.php?action=show&id=' . $id);
305 }
306
307 /**
308  * This method will update a row and make the record as deleted
309  *
310  * <pre>
311  * This method will take the id and set the delete_date field to
312  * the current datetime, which will marking it as deleted
313  *
314  * <div style="color:red">NOTE: This is generated information from the
315  * framework and will need to be corrected if there are any changes</div>
316  * </pre>
317  *
318  * @param Integer $id The primary id of the row to show
319  * @return Boolean success or failed
320  */
321
322 Private function remove($id){
323     if(empty($id)){
324         return false;
325     }
326
327     $sql = "UPDATE category SET delete_date=NOW() WHERE category_id =" . $id;
328
329     try{
330         $result = $this-> db-> update($sql);
331     }catch(CustomException $e){
332         echo $e-> queryError($sql);
333     }
334     return true;
335 }
336
337
338 /***** END CRUD METHOD*****/
339
340
341 /**
342  * Show list of information corresponding the to this class
343  *
344  * <pre>This Method will produce a list of all the element corresponding to the result of
345  * Category
346  * using the base/table.class.php file, which will format the list into a filtable table that
347  * uses ajax class to change the content on filtering
348  *
349  * There are to response type for this table for the parameter $type
350  *
351  * TABLE = Will return the content in a table with a filter row and a heading row
352  * AJAX = Will return just the content after evaluating the filter or heading information
353  *
354  * The parameter $filter expects an array with the key being the field to look for and the
355  * value being the the information to filter on
356  *
357  * <div style="color:red">NOTE: This is generated information from the
358  * framework and will need to be corrected if there are any changes</div>
359  * </pre>
360  *
361  * @param String $type Option of type of response for the output of the list
362  * @param String $orderby Which single field is used to order the output
363  * @param String $direction Which direction is required for the orderby output
364  * @param Array $filter A array of fields to filter, key=TEXT set (eg array('title'=>'this
365  * title'))
366  */
367
368 public function getCategoryList($type='TABLE', $orderby=NULL, $direction='ASC', $filter=NULL){
369
370     $result = $this-> lists($orderby, $direction, $filter);
371
372     $this-> table-> removeColumn(array('category_id'));
373 }

```

```

371         switch(strtoupper($type)){
372
373             case 'AJAX' : $this-> table-> setRowsOnly();
374                         $this-> table-> setIdentifier('catagory_id');
375                         $this-> table-> setIdentifierPage('category');
376                         echo $this-> table-> generateDisplayTable($result);
377
378             BREAK;
379             case 'TABLE' :
380             DEFAULT :
381                 $this-> table-> setHeader(array(
382                     'catagory_id'=> 'Category Id',
383                     'catagory_name'=> 'Category Name',
384                     'create_date'=> 'Create Date',
385                     'modify_date'=> 'Modify Date',
386                     'delete_date'=> 'Delete Date'));
387
388                 $this-> table-> setFilter(array(
389                     'catagory_id'=> 'TEXT',
390                     'catagory_name'=> 'TEXT',
391                     'create_date'=> 'TEXT',
392                     'modify_date'=> 'TEXT',
393                     'delete_date'=> 'TEXT'));
394
395                 $this-> table-> setIdentifier('catagory_id');
396
397                 $this-> template-> content(Box($this-> table->
> generateDisplayTable($result), 'Category List', 'Shows the current listings for the Category. To
create a new Listing <a href="category.php?action=create">Click Here</a>'
398                                     ));
399                 $this-> template-> display();
400             }
401         }
402
403         /**
404         * Show details of a single Category from the category
405         *
406         * <pre>This method will return a template page of the information requested
407         * the method use the template class to format the information ready to display the
408         * the user
409         *
410         * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
412         * </pre>
413         * @param Integer $id the primary id of the row to show
414         */
415         Public function showCategoryDetails($id){
416             $fieldMember = $this-> read($id);
417
418             $this-> template-> page('category.tpl.html');
419
420             $this-> templateCategoryLayout($fieldMember);
421
422             //if($this->checkAdminLevel(1)){
423                 $this-> template-> assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='category.php?action=edit&id="
424                                     . $id . "\">Edit</div>");
425             //}
426             echo $this-> template-> fetch();
427         }
428
429         /**
430         * Show the details ready to edit of a single Category from the category
431         *
432         * <pre>
433         * This method is used to display and editable page to the use, so that they
434         * maybe able to edit any of the fields releated to the row in question.
435         * The method uses the template class to format the information ready to display the
436         * the user
437         *
438         * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
440         * </pre>
441         * @param Integer $id The primary id of the row to show
442         */
443         Public function editCategoryDetails($id){
444

```

```

445         $fieldMember = $this-> read($id);
446
447         $name = 'editCategory';
448
449         $this-> template-> page('category.tpl.html');
450         $this-> template-> assign('FORM-HEADER', '<form
action="category.php?action=update&id=' . $id . '" method="POST"
name="' . $name . '">');
451
452         $this-> templateCategoryLayout($fieldMember, true);
453
454         $this-> template-> assign('FUNCTION', " <div class=\"button\"
onclick=\"document. $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='category.php?action=show&id= " . $id . "'>Cancel</div&
gt;");
455
456         $this-> template-> display();
457     }
458
459     /**
460      * update the information in a single Category from the category
461      *
462      * <pre>
463      * This method is used to take the information from editDepartmentDetails and try to validate
464      * it thought the validation method and on success will format it ready for input into the database through
465      * the update method
466      *
467      * if the validate fails then the user is show a page that mimics the editDepartmentDetails
468      * method and point out error in there input
469      *
470      * <div style="color:red">NOTE: This is generated information from the
471      * framework and will need to be corrected if there are any changes</div>
472      * </pre>
473      *
474      * @see editDepartmentDetails()
475      * @see Validate()
476      * @see update()
477      * @param Integer $id The primary id of the row to updated
478      */
479     Public function updateCategoryDetails($id){
480
481         if ($this-> Validate($_REQUEST)){
482
483             $request = $_REQUEST;
484             $table = 'category';
485
486             $save[$table]['catagory_name'] = $request['catagory_name'];
487
488             $save[$table]['modify_date'] = date('Y-m-d h:i:s');
489
490             $this-> update($save, $id);
491             header('Location: category.php?action=show&id=' . $id);
492         }else{
493
494             $fieldMember = $this-> valid_field;
495             $error = $this-> validation_error;
496
497             $name = 'editCategory';
498
499             $this-> template-> page('category.tpl.html');
500
501             foreach($error AS $key=> $value){
502                 $this-> template-> assign('err_' . $key, "<span
class=\"error\">" . @implode(' , ' , $error[$key]) . "</span>");
503             }
504
505             $this-> template-> assign('FORM-HEADER', '<form
action="category.php?action=update&id=' . $id . '" method="POST"
name="' . $name . '">');
506
507             $this-> templateCategoryLayout($fieldMember, true);
508
509             //if($this->admin->checkAdminLevel(1)){
510                 $this-> template-> assign('FUNCTION', " <div
class=\"button\" onclick=\"document. $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='category.php?action=show&id= " . $id . "'>Cancel</div&

```



```

509         );
510         $this-> template-> assign('FORM-FOOTER', '</form>' );
511
512         $this-> template-> display();
513     }
514 }
515
516 /**
517  * This method will provide a page to the to add a single row Category to the category table
518  *
519  * <pre>
520  * The method using the template class to format the information ready to display the
521  * the user, so that they may be able to add any of the fields releated to a row in the
522  database.
523  * The method uses the template class to format the information ready to display the
524  * the user
525  * <div style="color:red">NOTE: This is generated information from the
526  framework and will need to be corrected if there are any changes</div>
527  * </pre>
528  */
529 Public function createCategoryDetails(){
530     $name = 'createCategory';
531
532     $this-> template-> page('category.tpl.html');
533     $this-> template-> assign('FORM-HEADER', '<form
534 action="category.php?action=save" method="POST" name="' . $name . '">' );
535
536     $this-> templateCategoryLayout('', true);
537
538     $this-> template-> assign('FUNCTION', " <div class=\"button\"
539 onclick=\"document. $name.submit(); return false\">Save</div><div
540 class=\"button\"
541 onclick=\"location.href='category.php?action=list'\">Cancel</div> " );
542
543     $this-> template-> display();
544 }
545
546 /**
547  * save the information in a single Category to the category table
548  *
549  * <pre>
550  * This method is used to take the information from createDepartmentDetails and try to validate
551  it thought the
552  * validation method and on success will format it ready for inserted into the datebase through
553  the insert method
554  * if the validate fails then the user is show a page that mimics the createDepartmentDetails
555  method and point out
556  * error in there input
557  *
558  * <div style="color:red">NOTE: This is generated information from the
559  framework and will need to be corrected if there are any changes</div>
560  * </pre>
561  * @see createDepartmentDetails()
562  * @see Validate()
563  * @see update()
564  * @param Integer $id The primary id of the row to updated
565  */
566 Public function saveCategoryDetails(){
567     if ($this-> Validate($_REQUEST)){
568
569         $request = $_REQUEST;
570         $table = 'category';
571
572         $save[$table]['catagory_name'] = $request['catagory_name'];
573
574         $save[$table]['create_date'] = date('Y-m-d h:i:s');
575
576         $id = $this-> create($save);
577         header('Location: category.php?action=show&id=' . $id);
578     }else{
579
580         $fieldMember = $this-> valid_field;
581     }
582 }

```

```

578         $error = $this-> validation_error;
579
580         $name = 'createCategory';
581
582         $this-> template-> page('category.tpl.html');
583
584         foreach($error AS $key=> $value){
585             $this-> template-> assign('err_'. $key, "<span
class=\"error\">"
586                 .@implode(' ', $value). "</span>"
587             );
588
589             $this-> template-> assign('FORM-HEADER', '<form
action="category.php?action=save" method="POST" name="'
590                 . $name. '">'
591             );
592
593             $this-> templateCategoryLayout($fieldMember, true);
594
595             //if($this->admin->checkAdminLevel(1)){
596                 $this-> template-> assign('FUNCTION', " <div
class=\"button\" onclick=\"document.
597                 $name.submit(); return
false\">Update</div><div class=\"button\"
598                 onclick=\"location.href='category.php'>Cancel</div>
599                 "
600             );
601             $this-> template-> assign('FORM-FOOTER', '</form>'
602             );
603             $this-> template-> display();
604         }
605     }
606
607     /**
608     * Set a row to be marked as deleted
609     *
610     * <pre>
611     * This method will take the id and set the delete_date field to
612     * the current datetime, which will marking it as deleted
613     *
614     * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
615     * </pre>
616     * @param Integer $id The primary id of the row to marked as delete
617     */
618     public function deleteCategoryDetails($id){
619         $this-> remove($id);
620         header('Location: category.php');
621     }
622
623     /**
624     * This method assigns the associate array values to the template
625     *
626     * <pre>
627     * This method is used to incorporate the standards elements of the templates to a single
628     * function across all tempatied methods
629     * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
630     * </pre>
631     * @todo find out what $inputArray is used for
632     *
633     * @param Array $fielddata An associative array of fields that need to be assigned to the
template object
634     * @param Boolean $input If false then just assign the value if true the add the value to
corresponding form element
635     * @param Array $inputArray Not sure :S
636     */
637     private function templateCategoryLayout($fieldMember, $input = false, $inputArray=array()) {
638
639         $id = @$fieldMember['catagory_id'];
640
641         @$this-> template-> assign('catagory_id', ($input)? $this-
642 > template-> input('text', 'catagory_id', $fieldMember['catagory_id']):$fieldMember['catagory_id']);
643         @$this-> template-> assign('catagory_name', ($input)? $this-> template-
644 > input('text', 'catagory_name', $fieldMember['catagory_name']):$fieldMember['catagory_name']);
645         @$this-> template-> assign('create_date', ($input)? $this-> template-
646 > input('text', 'create_date', $fieldMember['create_date']):$fieldMember['create_date']);
647         @$this-> template-> assign('modify_date', ($input)? $this-> template-
648 > input('text', 'modify_date', $fieldMember['modify_date']):$fieldMember['modify_date']);
649         @$this-> template-> assign('delete_date', ($input)? $this-> template-
650 > input('text', 'delete_date', $fieldMember['delete_date']):$fieldMember['delete_date']);
651
652         /*if(isset($id)){
653             $this->template->assign('COMMENTS', $this->comment-

```

```

>getCommentBox($id, 'category'));
644         }*/
645     }
646 }
647
648
649 /**
650  * This medthod is used to validate inputs from form information
651  * <pre>
652  * This method will first check the if the fields are in the valid_field array and strip out
653  * any that are not
654  * Then it check that fields that require a value in them from the fields_required have a
655  * value, if not add an error to validation_error array
656  * Then it will check all the values to find out if the value match the type found in the
657  * fields_validation_type array, if not add an error to validation_error array
658  * <div style="color:red">NOTE: This is generated information from the
659  * framework and will need to be corrected if there are any changes</div>
660  * </pre>
661  * @see fields
662  * @see fields_required
663  * @see fields_validation_type
664  * @see validation_error
665  * @param Array $request
666 */
667 public function Validate($request){
668
669     unset($this-> valid_field);
670     unset($this-> validation_error);
671     $isvalid = True;
672
673     $validfields = $this-> fields;
674     $requiredfields = $this-> fields_required;
675     $fieldsvalidationtype = $this-> fields_validation_type;
676
677     foreach ($request AS $key=> $value){ //lets strip put unwanted or security violation
678         fields
679         if(in_array($key, $validfields)){
680             $this-> valid_field[$key] = $value; //pure fields
681         }
682     }
683
684     foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
685         though so we can check them, helps with checkboxes
686         if(!isset($this-> valid_field[$value])){
687             $this-> valid_field[$value] = '';
688         }
689     }
690
691     if(count($requiredfields) > 0 ){
692         foreach($requiredfields AS $value){ // lets check all the required fields have a value
693             if (empty($this-> valid_field[$value]) || $this-> valid_field[$value] ==
694             'NULL'){
695                 $this-> validation_error[$value][] = 'Field is Required'; //error field
696                 $isvalid = false;
697             }
698         }
699     }
700
701     //now lets validate
702     foreach ($this-> valid_field AS $key=> $value){
703         $value = trim($value);
704         if(!empty($value)){ // don't cheak if empty, alread done in required check
705
706             switch(@$fieldsvalidationtype[$key]){
707                 case 'TEXTAREA': if (strlen($value) > 1024) {
708                     $this-> validation_error[$key][] = 'Field longer then 1024
709                     characters';
710                     $isvalid = false;
711                 } break;
712                 case 'TEXT': if (strlen($value) > 1024) {
713                     $this-> validation_error[$key][] = 'Field longer then 1024
714                     characters';
715                     $isvalid = false;
716                 } break;
717                 case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
718                     $this-> validation_error[$key][] = 'not a valid SAP number';
719                 }
720             }
721         }
722     }
723 }

```

```

714         $isvalid = false;
715     } break;
716     case 'DECIMAL': if (!is_numeric($value)) {
717         $this-> validation_error[$key][] = 'Decimal value expected';
718         $isvalid = false;
719     } break;
720     case 'BOOL': if ((is_bool($value)) &&
721 (strtoupper($value)!="YES"
722 ) && ( $value != 1)) {
723         $this-> validation_error[$key][] = 'Please check';
724         $isvalid = false;
725     } break;
726     case 'INT': if (!is_numeric($value) && $value != 'NULL') {
727         $this-> validation_error[$key][] = 'Numeric value expected';
728         $isvalid = false;
729     } break;
730     case 'DATE': //$date = str_replace('/', '-', $value);
731                 //$date = str_replace("\\", '-', $date);
732                 @list($day, $month, $year) = explode('/', $value);
733                 if(!checkdate($month,$day, $year)){
734                     $this-> validation_error[$key][] = 'incorrect date
735 format, expecting dd/mm/yyyy';
736                 $isvalid = false;
737             } break;
738     case 'YEAR': if(!checkYear($value)){
739         $this-> validation_error[$key][] = 'incorrect year
740 format, expecting yyyy';
741         $isvalid = false;
742     } break;
743 }
744 }
745 }
746 return $isvalid;
747 }

```

# File Source for employmenttype.class.php

Documentation for this file is available at [employmenttype.class.php](http://www.phpdoc.org/package/PhpDocumentor/employmenttype.class.php)

```
1  <?php
2  /**
3   * Employmenttype Class
4   * <pre>
5   * This class is based on the table employmenttype
6   *
7   * <div style="color:red">NOTE: This is generated information from the framework
and will need to be corrected if there are any changes</div>
8   * </pre>
9   *
10  * @author Jennifer Erator <jason@lexxcom.com>
11  * @version 1.1 of the Framework generator
12  * @package PeopleScope
13  */
14
15  class employmenttype {
16
17      /**
18       * Connect to PDO object through database class
19       * @var Object
20       */
21      private $db_connect;
22
23      /**
24       * Database class object
25       * @var Object
26       */
27      private $db;
28
29      /**
30       * Table class object
31       * @var Object
32       */
33      public $table;
34
35      /**
36       * Template class object
37       * @var Object
38       */
39      public $template;
40
41      /**
42       * Array of field used in the database if not in this list is dropped from insert or update
43       * @var Array
44       */
45      private $fields =array('employmenttype_id', 'employmenttype', 'create_date', 'modify_date',
'delete_date');
46
47      /**
48       * Array of feilds require information when validating
49       * @var Array|null
50       */
51      private $fields_required =array('employmenttype');
52
53      /**
54       * Array of feilds and there types that are check when validating
55       * @var Array|null
56       */
57      private $fields_validation_type = array ('employmenttype_id'=> 'INT',
'employmenttype'=> 'TEXT', 'create_date'=> 'TEXT', 'modify_date'=> 'TEXT',
'delete_date'=> 'TEXT');
58
59      /**
60       * Array use to store any error found during Validation function
61       * @see Validation()
62       * @var Array
63       */
64  }
```

```

64     private $validation_error = array();
65
66     /**
67      * Constructor for this method
68      *
69      * <pre>
70      * The constructor will setup the required object for this class
71      * will initiate the database class, the table class and the template
72      * for this class to use
73      *
74      *
75      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
76      * </pre>
77      *
78      * @see db::
79      * @see table
80      * @see template
81      */
82     public function __construct() {
83         $this-> db = new db();
84
85         try {
86             $this-> db_connect = $this-> db-> dbh;
87         } catch (CustomException $e) {
88             $e-> logError();
89         }
90
91         $this-> table = new table();
92         $this-> template = new template();
93     }
94
95     /**
96      * Show will pull a list from the corresponding Employmenttype employmenttype
97      *
98      * <pre>
99      * This Method will produce a list of all the element corresponding to the result of
Employmenttype
100     Employmenttype
101     *
102     * I will only pull rows that are not considered delete
103     * eg. the delete_date field is not "0000-00-00 00:00:00" or set to NULL
104     *
105     * The parameter $filter expects an array with the key being the field to look for and the
106     * value being the the information to filter on
107     *
108     * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
109     * </pre>
110     *
111     * @param String $orderby Which single field is used to oder the output
112     * @param String $direction Which direction os required for the orderby output
113     * @param Array $filter A array of fields to filter, key=$val set (eg array('tile'=>'this
title'))
114     */
115     Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
116
117         $sql = "SELECT employmenttype_id,
118 employmenttype,
119 create_date,
120 modify_date,
121 delete_date FROM employmenttype WHERE (delete_date ='00-00-0000 00:00:00' OR delete_date IS
NULL)"
122         ;
123         if(is_array($filter)){
124             foreach($filter AS $key=> $value){
125                 if ($value != 'NULL' && !empty($value)){
126                     $sql .= " AND " . $value;
127                 }
128             }
129         }
130
131         if($orderby){
132             $sql .= " ORDER BY " . $orderby . " . $direction;
133         }
134
135         try{
136             $result = $this-> db-> select($sql);
137         }catch(CustomException $e){
138             echo $e-> queryError($sql);

```

```

139     }
140
141     return $result;
142 }
143
144 /**
145  * This method will take an array and insert it in the database
146  *
147  * <pre>
148  * This method will insert the formatted information into a database, the format for the array
149  * should be an associated array being the first key should be the table inserting with the keys
150  * for child array the fields that are being inserted too and the values to insert
151  *
152  * Array
153  * (
154  *     [users] => Array
155  *         (
156  *             [name] => Dave
157  *             [surname] => Smith
158  *             [email] => dave@dave.com
159  *         )
160  *     [staff] => Array
161  *         (
162  *             [staff_id] => 1245
163  *             [office_number] => 22
164  *             [drown_code] => bee223
165  *         )
166  * )
167  *
168  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
169  *
170  * </pre>
171  *
172  * @param Array $source
173  *
174  * @return Integer Return last inserted primary id
175  */
176 Private function create($source){
177     try{
178         $this-> db_connect-> beginTransaction();
179
180         foreach($source['employmenttype'] AS $key=> $val){
181             $field[] = $key;
182             $value[] = ":" . $key;
183         }
184
185         $sql = "INSERT INTO employmenttype (" . implode(' ', $field) . ") VALUES
(" . implode(' ', $value) . ");" ;
186
187         foreach($source['employmenttype'] AS $key=> $val){
188             $exec[":" . $key] = $val;
189         }
190
191         try{
192             $pid = $this-> db-> insert($sql, $exec);
193         }catch(CustomException $e){
194             throw new CustomException($e-> queryError($sql));
195         }
196         $this-> db_connect-> commit();
197     }
198
199     catch (CustomException $e) {
200         $e-> queryError($sql);
201         $this-> db_connect-> rollBack();
202         return false;
203     }
204
205     return $pid;
206 }
207
208
209 /**
210  * This method will return information as row
211  *
212  * <pre>
213  * This method is you to get a single row of information from the database
214  * based ith the primary id and return it as an array
215  *
216  * <div style="color:red">NOTE: This is generated information from the

```

```

framework and will need to be corrected if there are any changes</div>
217 * </pre>
218 *
219 * @param Integer $id The primary id of the row to show
220 */
221 Private function read($id){
222
223     $sql = "SELECT employmenttype_id,
224 employmenttype,
225 create_date,
226 modify_date,
227 delete_date FROM employmenttype WHERE employmenttype_id = " . $id . " AND (delete_date = '00-
00-0000 00:00:00' OR delete_date IS NULL)" ;
228
229
230     $stmt = $this-> db_connect-> prepare($sql);
231     $stmt-> execute();
232
233     try{
234         $result = $this-> db-> select($sql);
235     }catch(CustomException $e){
236         echo $e-> queryError($sql);
237     }
238
239     return $result[0];
240
241 }
242
243
244 /**
245  * This method will take an array and update a row in the database
246  *
247  * <pre>
248  * This method will update the formatted information into the database, the format for the array
249  * should be an associated array being the first key should be the table to be updated with the
250  * for child array the fields that are being updated too and the values to be updated
251  *
252  * Array
253  * (
254  *     [users] => Array
255  *         (
256  *             [name] => Dave
257  *             [surname] => Smith
258  *             [email] => dave@dave.com
259  *         )
260  *     [staff] => Array
261  *         (
262  *             [staff_id] => 1245
263  *             [office_number] => 22
264  *             [drown_code] => bee223
265  *         )
266  * )
267  *
268  * <div style="color:red">NOTE: This is generated information from the
269  * framework and will need to be corrected if there are any changes</div>
270  * </pre>
271  *
272  * @param Array $source
273  *
274  * @return Integer Return last inserted primary id
275  */
276 Private function update($source, $id){
277     try{
278         $this-> db_connect-> beginTransaction();
279
280         foreach($source['employmenttype'] AS $key=> $val){
281             $field[] = $key." = ":" . $key;
282         }
283
284         $sql = "UPDATE employmenttype SET " . implode(' , ', $field) . " WHERE
285 employmenttype_id = " . $id;
286
287         foreach($source['employmenttype'] AS $key=> $val){
288             $exec[":" . $key] = $val;
289         }
290
291         try{
292             $pid = $this-> db-> update($sql, $exec);

```



```

292         }catch(CustomException $e){
293             throw new CustomException($e-> queryError($sql));
294         }
295         $this-> db_connect-> commit();
296     }
297
298     catch (CustomException $e) {
299         $e-> queryError($sql);
300         $this-> db_connect-> rollBack();
301         return false;
302     }
303
304     header('Location:employmenttype.php?action=show&id=' . $id);
305 }
306
307 /**
308  * This method will update a row and make the record as deleted
309  *
310  * <pre>
311  * This method will take the id and set the delete_date field to
312  * the current datetime, which will marking it as deleted
313  *
314  * <div style="color:red">NOTE: This is generated information from the
315  * framework and will need to be corrected if there are any changes</div>
316  * </pre>
317  *
318  * @param Integer $id The primary id of the row to show
319  * @return Boolean success or failed
320  */
321
322 Private function remove($id){
323     if(empty($id)){
324         return false;
325     }
326
327     $sql = "UPDATE employmenttype SET delete_date=NOW() WHERE employmenttype_id
328     = " . $id;
329
330     try{
331         $result = $this-> db-> update($sql);
332     }catch(CustomException $e){
333         echo $e-> queryError($sql);
334     }
335     return true;
336 }
337
338 /***** END CRUD METHOD*****/
339
340 /**
341  * Show list of information corresponding the to this class
342  *
343  * <pre>This Method will produce a list of all the element corresponding to the result of
344  * Employmenttype
345  * using the base/table.class.php file, which will format the list into a filterable table that
346  * uses ajax class to change the content on filtering
347  *
348  * There are two response type for this table for the parameter $type
349  *
350  * TABLE = Will return the content in a table with a filter row and a heading row
351  * AJAX = Will return just the content after evaluating the filter or heading information
352  *
353  * The parameter $filter expects an array with the key being the field to look for and the
354  * value being the information to filter on
355  *
356  * <div style="color:red">NOTE: This is generated information from the
357  * framework and will need to be corrected if there are any changes</div>
358  * </pre>
359  *
360  * @param String $type Option of type of response for the output of the list
361  * @param String $orderby Which single field is used to order the output
362  * @param String $direction Which direction is required for the orderby output
363  * @param Array $filter A array of fields to filter, key=TEXT set (eg array('title'=>'this
364  * title'))
365  */
366
367 public function getEmploymenttypeList($type='TABLE', $orderby=NULL, $direction='ASC',
368 $filter=NULL){

```

```

366     $result = $this-> lists($orderby, $direction, $filter);
367
368
369     $this-> table-> removeColumn(array('employmenttype_id'));
370
371     switch(strtoupper($type)){
372
373         case 'AJAX' : $this-> table-> setRowsOnly();
374                     $this-> table-> setIdentifier('employmenttype_id');
375                     $this-> table-> setIdentifierPage('employmenttype');
376                     echo $this-> table-> generateDisplayTable($result);
377
378         BREAK;
379
380         case 'TABLE' :
381         DEFAULT :
382             $this-> table-> setHeader(array(
383                 'employmenttype_id'=> 'Employmenttype Id',
384                 'employmenttype'=> 'Employmenttype',
385                 'create_date'=> 'Create Date',
386                 'modify_date'=> 'Modify Date',
387                 'delete_date'=> 'Delete Date'));
388
389             $this-> table-> setFilter(array(
390                 'employmenttype_id'=> 'TEXT',
391                 'employmenttype'=> 'TEXT',
392                 'create_date'=> 'TEXT',
393                 'modify_date'=> 'TEXT',
394                 'delete_date'=> 'TEXT'));
395
396             $this-> table-> setIdentifier('employmenttype_id');
397
398             $this-> template-> content(Box($this-> table->
399 > generateDisplayTable($result), 'Employmenttype List', 'Shows the current listings for the
400 Employmenttype. To create a new Listing <a href="employmenttype.php?action=create">Click
401 Here</a>' ));
402
403             $this-> template-> display();
404         }
405     }
406
407     /**
408     * Show details of a single Employmenttype from the employmenttype
409     *
410     * <pre>This method will return a template page of the information requested
411     * the method use the template class to format the information ready to display the
412     * the user
413     *
414     * <div style="color:red">NOTE: This is generated information from the
415     framework and will need to be corrected if there are any changes</div>
416     * </pre>
417     * @param Integer $id the primary id of the row to show
418     */
419     public function showEmploymenttypeDetails($id){
420         $fieldMember = $this-> read($id);
421
422         $this-> template-> page('employmenttype.tpl.html');
423
424         $this-> templateEmploymenttypeLayout($fieldMember);
425
426         //if($this->checkAdminLevel(1)){
427             $this-> template-> assign('FUNCTION', "<div class=\"button\"
428 onclick=\"location.href='employmenttype.php?action=edit&id=\" . $id . \"\">Edit</
429 div>\"");
430         //}
431
432         echo $this-> template-> fetch();
433     }
434
435     /**
436     * Show the details ready to edit of a single Employmenttype from the employmenttype
437     *
438     * <pre>
439     * This method is used to display and editable page to the use, so that they
440     * maybe able to edit any of the fields releated to the row in question.
441     * The method uses the template class to format the information ready to display the
442     * the user
443     *
444     * <div style="color:red">NOTE: This is generated information from the

```

```

framework and will need to be corrected if there are any changes</div>
440      * </pre>
441      * @param Integer $id The primary id of the row to show
442      */
443      Public function editEmploymenttypeDetails($id){
444
445          $fieldMember = $this-> read($id);
446
447          $name = 'editEmploymenttype';
448
449          $this-> template-> page('employmenttype.tpl.html');
450          $this-> template-> assign('FORM-HEADER', '<form
action="employmenttype.php?action=update&id=' . $id . '" method="POST"
name=" ' . $name . '">' );
451
452          $this-> templateEmploymenttypeLayout($fieldMember, true);
453
454          $this-> template-> assign('FUNCTION', " <div class=\"button\"
onclick=\"document. $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='employmenttype.php?action=show&id= " . $id . "'>Cancel&lt
;/div>" );
455
456          $this-> template-> display();
457      }
458
459      /**
460      * update the information in a single Employmenttype from the employmenttype
461      *
462      * <pre>
463      * This method is used to take the information from editDepartmentDetails and try to validate
464      * it thought the
465      * validation method and on success will format it ready for input into the database through
466      * the update method
467      *
468      * if the validate fails then the user is show a page that mimics the editDepartmentDetails
469      * method and point out
470      * error in there input
471      *
472      * <div style="color:red">NOTE: This is generated information from the
473      * framework and will need to be corrected if there are any changes</div>
474      * </pre>
475      *
476      * @see editDepartmentDetails()
477      * @see Validate()
478      * @see update()
479      * @param Integer $id The primary id of the row to updated
480      */
481      Public function updateEmploymenttypeDetails($id){
482
483          if ($this-> Validate($_REQUEST)){
484
485              $request = $_REQUEST;
486              $table = 'employmenttype';
487
488              $save[$table]['employmenttype'] = $request['employmenttype'];
489
490              $save[$table]['modify_date'] = date('Y-m-d h:i:s');
491
492              $this-> update($save, $id );
493              header('Location: employmenttype.php?action=show&id=' . $id);
494          }else{
495
496              $fieldMember = $this-> valid_field;
497              $error = $this-> validation_error;
498
499              $name = 'editEmploymenttype';
500
501              $this-> template-> page('employmenttype.tpl.html');
502
503              foreach($error AS $key=> $value){
504                  $this-> template-> assign('err_' . $key, "<span
class=\"error\">" . implode(', ', $error[$key]) . "</span>" );
505              }
506
507              $this-> template-> assign('FORM-HEADER', '<form
action="employmenttype.php?action=update&id=' . $id . '" method="POST"
name=" ' . $name . '">' );
508
509              $this-> templateEmploymenttypeLayout($fieldMember, true);

```

```

506
507         //if($this->admin->checkAdminLevel(1)){
508             $this-> template-> assign('FUNCTION', "      <div
class=\"button\" onclick=\"document.
false\">Update</div><div class=\"button\"
onclick=\"location.href='employmenttype.php?action=show&id=
\" . $id. \"'>Cancel&lt
;/div>"
        );
509         //}
510         $this-> template-> assign('FORM-FOOTER', '</form>'
        );
511
512         $this-> template-> display();
513     }
514 }
515
516 /**
517  * This method will provide a page to the to add a single row Employmenttype to the
employmenttype table
518  *
519  * <pre>
520  * The method using the template class to format the information ready to display the
521  * the user, so that they may be able to add any of the fields related to a row in the
database.
522  * The method uses the template class to format the information ready to display the
523  * the user
524  *
525  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
526  * </pre>
527  */
528 Public function createEmploymenttypeDetails(){
529
530     $name = 'createEmploymenttype';
531
532     $this-> template-> page('employmenttype.tpl.html');
533     $this-> template-> assign('FORM-HEADER', '<form
action="employmenttype.php?action=save" method="POST"
name="' . $name. '">'
        );
534
535     $this-> templateEmploymenttypeLayout('', true);
536
537     $this-> template-> assign('FUNCTION', "      <div class=\"button\"
onclick=\"document.
class=\"button\"
onclick=\"location.href='employmenttype.php?action=list'>Cancel</div>
"
        );
538
539
540     $this-> template-> display();
541 }
542
543 /**
544  * save the information in a single Employmenttype to the employmenttype table
545  *
546  * <pre>
547  * This method is used to take the information from createDepartmentDetails and try to validate
it thought the
548  * validation method and on success will format it ready for inserted into the database through
the insert method
549  *
550  * if the validate fails then the user is show a page that mimics the createDepartmentDetails
method and point out
551  * error in there input
552  *
553  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
554  * </pre>
555  *
556  * @see createDepartmentDetails()
557  * @see Validate()
558  * @see update()
559  * @param Integer $id The primary id of the row to updated
560  */
561 Public function saveEmploymenttypeDetails(){
562
563     if ($this-> Validate($_REQUEST)){
564
565         $request = $_REQUEST;
566         $table = 'employmenttype';
567
568         $save[$table]['employmenttype'] = $request['employmenttype'];
569

```

```

570         $save[$table]['create_date'] = date('Y-m-d h:i:s');
571
572         $id = $this-> create($save);
573         header('Location: employmenttype.php?action=show&id=' . $id);
574     }else{
575
576         $fieldMember = $this-> valid_field;
577
578         $error = $this-> validation_error;
579
580         $name = 'createEmploymenttype';
581
582         $this-> template-> page('employmenttype.tpl.html');
583
584         foreach($error AS $key=> $value){
585             $this-> template-> assign('err_'. $key, "<span
class=\"error\">" . @implode(' ', $value) . "</span>" );
586         }
587
588         $this-> template-> assign('FORM-HEADER', '<form
action="employmenttype.php?action=save" method="POST"
name="' . $name . '">');
589
590         $this-> templateEmploymenttypeLayout($fieldMember, true);
591
592         //if($this->admin->checkAdminLevel(1)){
593             $this-> template-> assign('FUNCTION', " <div
class=\"button\" onclick=\"document. $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='employmenttype.php'>Cancel</div> " );
594         //}
595         $this-> template-> assign('FORM-FOOTER', '</form>');
596
597         $this-> template-> display();
598     }
599 }
600
601 /**
602  * Set a row to be marked as deleted
603  *
604  * <pre>
605  * This method will take the id and set the delete_date field to
606  * the current datetime, which will marking it as deleted
607  *
608  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
609  * </pre>
610  * @param Integer $id The primary id of the row to marked as delete
611  */
612 Public function deleteEmploymenttypeDetails($id){
613     $this-> remove($id);
614     header('Location: employmenttype.php');
615 }
616
617 /**
618  * This method assigns the associate array values to the template
619  *
620  * <pre>
621  * This method is used to incorporate the standards elements of the templates to a single
622  * function across all tempatled methods
623  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
624  * </pre>
625  * @todo find out what $inputArray is used for
626  *
627  * @param Array $fielddata An associative array of fields that need to be assigned to the
template object
628  * @param Boolean $input If false then just assign the value if true the add the value to
corresponding form element
629  * @param Array $inputArray Not sure :S
630  */
631 private function templateEmploymenttypeLayout($fieldMember, $input = false, $inputArray=array())
632 {
633     $id = @$fieldMember['employmenttype_id'];
634
635     @$this-> template-> assign('employmenttype_id', ($input)? $this-
> template-> input('text', 'employmenttype_id',
636     $fieldMember['employmenttype_id']):$fieldMember['employmenttype_id']);
637     @$this-> template-> assign('employmenttype', ($input)? $this-> template-

```

```

> input('text', 'employmenttype', $fieldMember['employmenttype']):$fieldMember['employmenttype']);
637 @<this-> template-> assign('create_date', ($input)? $this-> template-
> input('text', 'create_date', $fieldMember['create_date']):$fieldMember['create_date']);
638 @<this-> template-> assign('modify_date', ($input)? $this-> template-
> input('text', 'modify_date', $fieldMember['modify_date']):$fieldMember['modify_date']);
639 @<this-> template-> assign('delete_date', ($input)? $this-> template-
> input('text', 'delete_date', $fieldMember['delete_date']):$fieldMember['delete_date']);
640
641
642 /*if(isset($id)){
643     $this->template->assign('COMMENTS', $this->comment-
>getCommentBox($id, 'employmenttype'));
644 }*/
645
646 }
647
648
649 /**
650  * This method is used to validate inputs from form information
651  * <pre>
652  * This method will first check the if the fields are in the valid_field array and strip out
any that are not
653  * Then it check that fields that require a value in them from the fields_required have a
value, if not add an error to validation_error array
654  * Then it will check all the values to find out if the value match the type found in the
fields_validation_type array, if not add an error to validation_error array
655  * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
656  * </pre>
657  *
658  * @see fields
659  * @see fields_required
660  * @see fields_validation_type
661  * @see validation_error
662  *
663  * @param Array $request
664  */
665 public function Validate($request){
666
667     unset($this-> valid_field);
668     unset($this-> validation_error);
669     $isvalid = True;
670
671     $validfields = $this-> fields;
672     $requiredfields = $this-> fields_required;
673     $fieldsvalidationtype = $this-> fields_validation_type;
674
675     foreach ($request AS $key=> $value){ //lets strip put unwanted or security violation
fields
676         if(in_array($key, $validfields)){
677             $this-> valid_field[$key] = $value; //pure fields
678         }
679     }
680
681     foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxes
682         if(!isset($this-> valid_field[$value])){
683             $this-> valid_field[$value] = '';
684         }
685     }
686
687     if(count($requiredfields) > 0 ){
688         foreach($requiredfields AS $value){ // lets check all the required fields have a value
689             if (empty($this-> valid_field[$value]) || $this-> valid_field[$value] ==
'NULL'){
690                 $this-> validation_error[$value][] = 'Field is Required'; //error field
691                 $isvalid = false;
692             }
693         }
694     }
695
696
697     //now lets validate
698     foreach ($this-> valid_field AS $key=> $value){
699         $value = trim($value);
700         if(!empty($value)){ // don't check if empty, already done in required check
701
702             switch(@$fieldsvalidationtype[$key]){
703                 case 'TEXTAREA': if (strlen($value) > 1024) {

```

```

705             $this-> validation_error[$key][] = 'Field longer then 1024
characters';
706             $isvalid = false;
707         } break;
708     case 'TEXT': if (strlen($value) > 1024) {
709         $this-> validation_error[$key][] = 'Field longer then 1024
characters';
710         $isvalid = false;
711     } break;
712     case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
713         $this-> validation_error[$key][] = 'not a valid SAP number';
714         $isvalid = false;
715     } break;
716     case 'DECIMAL': if (!is_numeric($value)) {
717         $this-> validation_error[$key][] = 'Decimal value expected';
718         $isvalid = false;
719     } break;
720     case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"
) && ( $value != 1)) {
721         $this-> validation_error[$key][] = 'Please check';
722         $isvalid = false;
723     } break;
724     case 'INT': if (!is_numeric($value) && $value != 'NULL' ){
725         $this-> validation_error[$key][] = 'Numeric value expected';
726         $isvalid = false;
727     } break;
728     case 'DATE': //$date = str_replace('/', '-', $value);
729                 //$date = str_replace("\\", '-', $date);
730                 @list($day, $month, $year) = explode('/', $value);
731                 if(!checkdate($month,$day, $year)){
732                     $this-> validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
733                     $isvalid = false;
734                 } break;
735     case 'YEAR': if(!checkYear($value)){
736         $this-> validation_error[$key][] = 'incorrect year
format, expecting yyyy';
737         $isvalid = false;
738     } break;
739
740     }
741 }
742 }
743
744 return $isvalid;
745
746 }
747 }

```

# File Source for database.class.php

Documentation for this file is available at [database.class.php](http://www.phpdoc.org/package/PhpDocumentor/database.class.php)

```
1  <?php
2  /**
3   * Database Class,
4   * <br />
5   * This class is Database abstraction layer<br />
6   *
7   * Example:<br />
8   * <br />
9   * define('DB_USER','root');<br />
10  * define('DB_PASS','password');<br />
11  * define('DB_HOST','localhost');<br />
12  * define('DB_DBASE','test_db');<br />
13  * define('DB_TYPE','mysql');<br />
14  * <br />
15  * try{$db = new db();}<br />
16  * catch(CustomException e){ echo $e->logError(); }<br />
17  * <br />
18  * try{$result = $db->select('select * from test_table');}<br />
19  * catch(CustomException e){echo $e->queryError();}<br />
20  * <br />
21  * foreach($result AS $row){<br />
22  *     echo 'col1 =' . $row['col1'];<br />
23  *     echo 'col2 =' . $row['col2'];<br />
24  *     echo 'col3 =' . $row['col3'];<br />
25  * }<br />
26  * <br />
27  *
28  * @author Jason Stewart <jason@lexxcom.com.au>
29  * @version 1.0
30  * @package PeopleScope
31  * @subpackage Base
32  */
33
34  if (isset($_SESSION['dbaccess']['server_host'])) {
35      $DB_HOST = $_SESSION['dbaccess']['server_host'];
36  }else{
37      $DB_HOST = DB_HOST;
38  }
39
40  if (isset($_SESSION['dbaccess']['server_database'])) {
41      $DB_DBASE = $_SESSION['dbaccess']['server_database'];
42  }else{
43      $DB_DBASE = DB_DBASE;
44  }
45
46  if (isset($_SESSION['dbaccess']['server_type'])) {
47      $DB_TYPE = strtolower($_SESSION['dbaccess']['server_type']);
48  }else{
49      $DB_TYPE = DB_TYPE;
50  }
51
52  if (isset($_SESSION['dbaccess']['server_port'])) {
53      $DB_PORT = $_SESSION['dbaccess']['server_port'];
54  }else{
55      $DB_PORT = DB_PORT;
56  }
57
58  /**
59   * required general tool
60   */
61  require_once 'tools.function.php';
62  /**
63   * required error handler
64   */
65  require_once 'errorhandler.class.php';
66
67
```



```

68  /**
69  * This class is Database abstraction layer
70  *
71  * @package PeopleScope
72  * @subpackage Base
73  */
74  class db {
75
76      var $dsn;
77      var $dbh;
78      var $lastQuery;
79
80
81      /**
82       * constructor to initiate database conection
83       *
84       * @return Void
85       */
86      function __construct(){
87
88          $this-> dsn = DB_TYPE.':dbname='.DB_DBASE.':host='.DB_HOST.':port='.DB_PORT;
89          try{
90              try {
91                  $this-> dbh = new PDO($this-> dsn, DB_USER, DB_PASS);
92              } catch (PDOException $e) {
93                  throw new CustomException('Connection Error '. $e-> getMessage());
94              }
95          }catch(CustomException $e) {
96              echo $e-> logError();
97          }
98      }
99
100     /**
101      * returns current DSN string used to connect ot the server
102      *
103      * @return String
104      */
105     public function getDNSString(){
106         return $this-> dsn;
107     }
108
109     /**
110      * Will instigate a SELECT query and return and an array of responses
111      *
112      * @param String $sql select sql string to be executed
113      * @return Array
114      */
115     public function select($sql){
116
117         try{
118             $stmt = $this-> query($sql);
119         }catch(CustomException $e){
120             throw new CustomException('SELECT : '. $e-> getMessage());
121         }
122
123         return $stmt-> fetchAll(PDO::FETCH_NAMED);
124     }
125
126
127     /**
128      * Will instigate a INSERT query and return the inserts autocomplete Id;
129      *
130      * @param String $sql
131      * @return Array
132      */
133     public function insert($sql, $exec = NULL){
134
135         try{
136             $stmt = $this-> query($sql, $exec);
137         }catch(CustomException $e){
138             throw new CustomException('INSERT : '. $e-> getMessage());
139         }
140
141         return $this-> dbh-> lastInsertId();
142     }
143
144
145     /**
146      * Will instigate a DELETE query and return true if no problems;
147      *

```

```

148     * @param string $sql
149     * @return Array
150     */
151     public function delete($sql){
152
153         try{
154             $stmt = $this-> query($sql);
155         }catch(CustomException $e){
156             throw new CustomException('DELETE : '. $e-> getMessage());
157         }
158
159         return true;
160     }
161
162     /**
163     * Will instigate a UPDATE query and return true if no problems;
164     *
165     * @param string $sql
166     * @return array
167     */
168     public function update($sql, $exec=NULL){
169
170         try{
171             $stmt = $this-> query($sql, $exec);
172         }catch(CustomException $e){
173             throw new CustomException('UPDATE : '. $e-> getMessage());
174         }
175
176         return true;
177     }
178
179     /**
180     * Will instigate a query and return a recordset;
181     *
182     * @param string $sql
183     * @return array
184     */
185     public function query($sql, $exec=NULL){
186         $this-> lastQuery = $sql;
187
188         try{
189             $stmt = $this-> dbh-> prepare($sql);
190         }catch(CustomException $e){
191             throw new CustomException('QUERY : '. $e-> getMessage());
192         }
193
194         if(empty($exec)){
195             $stmt-> execute();
196         }else{
197             $stmt-> execute($exec);
198         }
199
200         if ($stmt-> errorCode() != 00000 )
201         {
202             if(empty($exec)){
203                 $error = $stmt-> errorInfo();
204             }else{
205                 $error = $stmt-> errorInfo();
206                 ob_start();
207                 $stmt-> debugDumpParams();
208                 $error[2] .= "<br /><pre>"
209 ob_get_contents()."</pre>" ;
210                 ob_end_clean();
211             }
212             // $error = $stmt->errorInfo();
213             throw new CustomException($error[2]);
214         }
215
216         return $stmt;
217     }
218
219     /**
220     * Merge query template with array
221     *
222     * Query template and array set merger function
223     * Will taken in the Query string template and the array of query elements
224     * and combine the to show the fully converted string used in the prepare
225     * Note: only use as example for debugging purposes
226     *
227     * @param string $query Query template

```

```

227      * @param array $params Array of inputs
228      * @return string
229      */
230      public function prepareToQuery($query, $params){
231          $keys = array();
232
233          # build a regular expression for each parameter
234          foreach ($params as $key => $value) {
235              if (is_string($key)) {
236                  $keys[] = '/' . $key . '/';
237              } else {
238                  $keys[] = '/[?]/';
239              }
240          }
241
242          $query = preg_replace($keys, $params, $query, 1, $count);
243
244          return $query;
245      }
246  }
247 }
248
249 ?>

```

# File Source for email.class.php

Documentation for this file is available at [email.class.php](http://pear.php.net/package/PhpDocumentor)

```
1  <?php
2  /**
3   * Email Class,
4   * <br />
5   * This class is used create and send email, can also record emails in a db, and can use the
template class to format Html <br />
6   *
7   *
8   * @author Jason Stewart <jason@lexxcom.com.au>
9   * @version 1.0
10  * @package PeopleScope
11  * @subpackage Base
12  */
13
14  /**
15   * Pear Mail class
16   */
17  require_once ('Mail.php');
18  /**
19   * Pear Mime class
20   */
21  require_once ('Mail/mime.php');
22
23  /**
24   * This class is used create and send email<br />
25   *
26   * @package PeopleScope
27   * @subpackage Base
28   */
29  class email{
30
31      /**
32       * Carrage return
33       * @var String
34       */
35      var $crlf = "\n" ;
36
37      /**
38       * Set Mime Type
39       * @var unknown_type
40       */
41      var $mime;
42
43      /**
44       * Mail Object
45       * @var Object
46       */
47      var $mail;
48
49      /**
50       * Template Object
51       * @var Object
52       */
53      var $template;
54
55      /**
56       * Mail Header information
57       * @var Araay
58       */
59      var $header;
60
61      /**
62       * Constructor for Email php5
63       *
64       * @param unknown_type $type
65       */
66      function __construct(){
```

```

67         $this-> email();
68     }
69
70     /**
71      * Constructor for Email.php4
72      *
73      */
74     function email(){
75         $this-> mime = new Mail_mime($this-> crlf);
76         if (PEAR::isError($this-> mime)) { pp($this-> mime-> getMessage(),1);}
77     }
78
79     /**
80      * converts the HTML version of the email to be sent to be appended to the email
81      *
82      * @param String $html
83      */
84     function append_html($html) {
85         $this-> mime-> setHTMLBody($html);
86     }
87
88     /**
89      * converts the TEXT version of the email to be sent to be appended to the email
90      *
91      * @param String $text
92      */
93     function append_text($text) {
94         $this-> mime-> setTXTBody($text);
95     }
96
97     /**
98      * converts the TEXT version of the email to be sent to be appended to the email
99      *
100     * @param String $file file path to the, or file content
101     * @param String $type Mime type of the thaile attached
102     */
103     function append_file($file, $type){
104         $this-> mime-> addAttachment($file, $type);
105     }
106
107     /**
108      * Create a HTML email using the base template class
109      *
110     * @param String $template path to template
111     * @param String $content array of assigned variable conversion
112     * @see template::assignArray
113     */
114     function setHTMLtemplate($template, $content){
115         $this-> template = new template;
116         $this-> template-> set($template);
117         $this-> template-> assignArray($content);
118         $this-> mime-> setHTMLBody($this-> template-> fetch());
119     }
120
121     /**
122      * Will create and output from Form Class standardised format String
123      * To create a TEXT email in a reable format
124      *
125     * @todo confirm this is correct
126     * @param String $table Form Class standardised format String
127     * @param String $content Content
128     */
129     function setTXTtemplate($table,$content){
130         $table = trim($table);
131         $rows = explode("\n" , $table);
132
133         $sstr = '';
134         foreach ($rows AS $value){
135             if ($value == "-----" OR $value == "\n" ){
136                 next();
137             }
138             //Nulled Error as as some feilds may not need validating
139             @list($info,$validation) = explode(':', $value);
140             list($lable,$ffield) = explode(':', $info);
141
142             $sstr .= $lable. " = " ;
143             if (isset($content[$ffield])){
144                 if (is_array($content[$ffield])){
145                     $sstr .= implode(", " , $content[$ffield]) . "\n" ;
146                 }else{

```

```

147         $sstr .= $content[$field] . "\n" ;
148     }
149 }
150 }
151
152 $this-> mime-> setTXTBody($sstr);
153 }
154
155 /**
156  * Set the Sender info for header
157  *
158  * @param String $from email of the sender
159  */
160 function sender($from){
161     $this-> header['From'] = $from;
162 }
163
164 /**
165  * Set the Subject line fo the email
166  * @param String $subject
167  */
168 function subject($subject){
169     //$this->header['Subject'] = $subject;
170     $this-> mime-> setSubject($subject);
171 }
172
173 /**
174  * Set all other Header information as required
175  *
176  * @param Array $header header param to header value
177  */
178 function setHeader($header){
179     if (!empty($header)){
180         if (is_array($header)){
181             $this-> header = $header;
182             return true;
183         }
184     }
185
186     return false;
187 }
188
189 /**
190  * Record email into the database
191  *
192  * @param String $table table to store the email
193  * @param String $content content of the email
194  * @param String $form_name identifier of record, Default URI sent from
195  */
196 function recordEmail($table, $content, $form_name = NULL ){
197
198     $this-> dbase = global_db();
199
200     if(empty($form_name)){
201         $form_name = $_SERVER["REQUEST_URI"] ;
202     }
203
204     $create_date = dbase_date_format();
205
206     //$set_fields = "sent_to, subject_line, form_name, creat_date";
207     $set_values = array("sent_to" => $this-> lastsent ,
208 "subject_line" => $this-> header['Subject'], "form_name" => $form_name,
209 "create_date" => $create_date);
210
211     $contentValueArray = array_merge($content, $set_values);
212     $fieldcount = count($contentValueArray);
213     $columnsArray = array_keys($contentValueArray);
214     $columnStr = implode(',', $columnsArray);
215
216     $sql = "INSERT INTO " . $table . " (" . $columnStr . ") values
217 (" . str_repeat ( "? , " , $fieldcount-1 )."?);" ;
218
219     $recordinsert = $this-> dbase-> Prepare($sql);
220     $ok = $this-> dbase-> Execute($recordinsert, $contentValueArray);
221
222     if (!$ok) {
223         pp($this-> dbase-> ErrorMsg(),1);
224         return false;
225     }
226 }
227

```

```

224         return true;
225     }
226
227     /**
228      * compile and Send email
229      *
230      * @param String $to Email address to send too;
231      */
232     function send($to){
233
234         $this->    lastsent = $to;
235         $body = $this->    mime->    get();
236         $hdrs = $this->    mime->    headers($this->    header);
237         $mail = Mail::factory(MAILTYPE,array('debug' =>    true));
238         if (PEAR::isError($this->    mail)) { return pp("Error:" . $this->    mail-
> getMessage());}
239         return $mail->    send($this->    lastsent, $hdrs, $body);
240     }
241 }
242 ?>

```

# File Source for form.class.php

Documentation for this file is available at [form.class.php](http://www.phpdoc.org/package/PhpDocumentor)

```
1  <?php
2
3  /**
4   * Form Class,
5   * <br />
6   * This class is used to take a input String in a standardised format and convert to a Html Form
7  <br />
8   *
9   * Example:<br />
10  * <br />
11  * $formVars = "<br />
12  * Title*:1:select:Ms|Mrs|Mrs, Miss|Miss, Mr|Mr::required:notype<br />
13  * Home Phone:5:editor::400|400::optional:notype<br />
14  * First Name*:2:text:::required:notype<br />
15  * Last Name*:3:text:::required:notype<br />
16  * Work Phone:4:text:::optional:notype<br />
17  * Home Phone:5:text:::optional:notype<br />
18  * Mobile Phone*:6:text:::required:notype<br />
19  * Email*:7:text:::required:notype<br />
20  * Address:8:textarea:::optional:notype<br />
21  * Suburb:9:text:::optional:notype";<br />
22  * <br />
23  * <br />
24  * $form = new form($formVars);<br />
25  * $formVal = $form->formHeader('');<br />
26  * $formVal .= $form->draw();<br />
27  * $formVal .= $form->freeFormSubmit();<br />
28  * $formVal .= $form->formFooter();<br />
29  * <br />
30  * echo $formVal<br />
31  *
32  * @author Jason Stewart <jason@lexxcom.com.au>
33  * @version 1.0
34  * @package PeopleScope
35  * @subpackage Base
36  */
37
38  /**
39   * This class is used to take a input String in a standardised format and convert to a Html Form
40   *
41   * @package PeopleScope
42   * @subpackage Base
43   */
44  class form extends table {
45
46      /**
47       * Database object
48       * @var Object
49       */
50      var $db;
51
52      /**
53       * Holds string value for form builder
54       * @var String
55       */
56      var $form = "" ;
57
58      /**
59       * Form enctype type
60       * @var String
61       */
62      var $enctype;
63
64      /**
65       * Java scripts reloaded to this class
66       * @var String
67       */
68  }
```



```

67     var $formScript;
68
69     /**
70      * Width size of fck editor
71      * @var Integer
72      */
73     var $fck_width = 300;
74
75     /**
76      * Height size of fck editor
77      * @var Integer
78      */
79     var $fck_height = 500;
80
81     //var $template;
82     /**
83      * JavaScript handle onChange for Form input types command
84      *
85      * @todo should be removed for JQuery ready({})
86      * @var String
87      */
88     var $autoRefresh;
89
90     /**
91      * JavaScript handle onClick for Form input types command
92      * use this for check boxes and radio
93      * @todo should be removed for JQuery ready({})
94      * @var String
95      */
96     var $autoRefreshcheckboxs;
97
98     /**
99      * JavaScript handle onkeyup for Form input types command
100     * user this to to action on an text input with an enter button
101     * @todo should be removed for JQuery ready({})
102     * @var String
103     */
104     var $autoRefreshInputEnter;
105
106     /**
107     * Url Tor find fck config file
108     * @var String
109     */
110     var $fck_configUrl;
111
112     /**
113     * Constrtuctor
114     *
115     * @param String $form input values for form information
116     * @return void
117     */
118     function __construct($form){
119         $this-> form($form);
120     }
121
122     /**
123     * Constrtuctor php4
124     *
125     * @param String $form input values for form information
126     * @return void
127     */
128     function form($form){
129         global $global_db;
130         $this-> db = $global_db;
131         $this-> form = trim($form);
132         //$this->template= new template();
133         $this-> fck_configUrl = SITE_ROOT."config/fckconfig.js" ;
134     }
135
136     /**
137     * Set onSubmit Form scripts
138     *
139     * @todo should be removed for JQuery ready({})
140     * @param String $script Javascript command or function
141     * @return void
142     */
143     function formScript($script){
144         $this-> formScript = $script;
145     }
146

```

```

147  /**
148  * Standards Submit Button
149  * @param String $value Button lable/Name
150  * @return void
151  */
152  function freeFormSubmit($value = 'Submit'){
153      return '<input type="submit" value="' . $value . '" />' ;
154  }
155
156  /**
157  * Setup the form tag ready for the inputs
158  *
159  * @param String $action Form action parameter
160  * @param String $method Form Method parameter
161  * @param String $formname Form Name and Id parameter
162  * @return string
163  */
164  function formHeader($action, $method=NULL, $formname=NULL){
165      //find any reference uploading file and set form accordingly
166      $columnBreakDown = explode("\n" , $this-> form);
167
168      foreach ($columnBreakDown AS $value){
169          $row = explode(":" , $value);
170          if ('upload' == trim(strtolower($row[2]))){
171              $this-> enctype="multipart/form-data" ;
172              $method = "POST" ;
173          }
174      }
175
176      //check if method is either GET or POST if not then set POST as default
177      $method = trim(strtoupper($method));
178      if($method != "POST" && $method != "GET" ){
179          $method = "POST" ;
180      }
181
182      //remove any spaces from $formname as can not be used with spaces
183      $formname = str_replace(" " , '' , $formname);
184
185      $ret = '<form method="' . $method . '" action="' . $action . '"';
186      $ret .= (!empty($formname)) ? ' name="' . $formname . '" id="' . $formname . '"';
187      : '';
188      $ret .= ($this-> enctype) ? ' enctype="' . $this-> enctype . '" : '' ;
189      $ret .= ($this-> formScript) ? ' onsubmit="' . $this-> formScript . '" : '' ;
190      $ret .= '>' ;
191
192      return $ret;
193  }
194
195  /**
196  * Setup closing form tag
197  *
198  * @return String
199  */
200  function formFooter(){
201      $ret = '</form>' ;
202
203      return $ret;
204  }
205
206  /**
207  * Will Generated the required input fields from the $this->form information
208  * information can be broken up using the string '-----'
209  *
210  * @param Integer $column Define which column should be returned
211  * @return String Html version on the generated input fields
212  */
213  function draw($column = NULL){
214
215      if($column){
216          $this-> form = str_replace("-----\r\n" , "-----" , $this->
> form);
217          $this-> form = str_replace("-----\r" , "-----" , $this-> form);
218          $this-> form = str_replace("-----\r" , "-----" , $this-> form);
219          $columnList = explode("-----" , $this-> form);
220          $column -= 1;
221      }else{
222          $columnList[0] = str_replace("-----\r\n" , '' , $this-> form);
223          $column = 0;
224      }

```

```

225     $count=0;
226     //Null Error
227     $columnBreakDown = @explode("\n" , $columnList[$column]);
228
229     foreach($columnBreakDown AS $value){
230         if (!empty($value)){
231             $inputfield[] = $this-> inputfield($value);
232         }
233     }
234     return @implode("\n" , $inputfield);
235 }
236
237 /**
238  * Set a alternate FCK config file path
239  *
240  * @param String $url Url to new config file
241  * @return void
242  */
243 function SetFckConfigUrl($url){
244     $this-> fck_configUrl = $url;
245 }
246
247 /**
248  * Takes in a standard single row from $this->form information and
249  * generated and html form input string
250  *
251  * @param String $input string eg.
252  State:10:select:ACT|ACT,NSW|NSW,VIC|VIC,QLD|QLD,SA|SA,NT|NT,WA|WA,TAS|TAS::optional:notype
253  * @return Html form input
254  */
255 function inputfield($input){
256     $fieldData = explode(":::" , $input);
257     //setting vars
258     $file = '';
259     $funcVars = '';
260     $func_div = false;
261     $isdiv = false;
262     $nolable = false;
263     $valid_str = '';
264     $func_script = '';
265     $func_class = '';
266     $func_id = '';
267
268     if(preg_match( "/^:/" ,@$fieldData[1])){
269         $fieldData[1] = substr($fieldData[1], 1);
270     }
271     $field = explode(':', trim($fieldData[0]));
272
273     $valid_str = '';
274     if(!empty($fieldData[1])){
275         $valid = explode(':', trim($fieldData[1]));
276         $valid_str = " validitycheck=\" $field[1];$field[0];$valid[0];$valid[1];0;-
277 1\"
278 ";
279     }else{
280         @$valid_str = " validitycheck=\" $field[1];$field[0];optional:notype;0;-
281 1\"
282 ";
283     }
284     if (!isset($field[3])){
285         $field[3]=''; // if value not set then create and set to blank
286     }
287     //nullled Error as value can be blank in this case
288     if (@isset($field[4])){
289         $func = explode(',', trim($field[4]));
290
291         foreach($func AS $value){
292             if(preg_match( "/^J|/" , $value)){
293                 $value=str_replace("J|" , '',$value);
294                 $func_script = trim($value)." " ;
295             }
296             if(preg_match( "/^C|/" , $value)){
297                 $value=str_replace("C|" , '',$value);
298                 $func_class = "class=" .trim($value)." " ;
299             }
300             if(preg_match( "/^I|/" , $value)){
301                 $value=str_replace("I|" , '',$value);
302                 $func_id = "id=" .trim($value)." " ;
303             }
304             if(preg_match( "/^D|/" , $value)){

```

```

302         $value=str_replace("D|" ,'', $value);
303         $func_div = "<div id=" .trim($value).">" ;
304     }
305     if(preg_match( "/^L|/" , $value)){
306         $value=str_replace("L|" ,'', $value);
307         if ($value == "NO_LABEL" ) $nolable = true;
308     }
309 }
310
311 $funcVars = $func_script.$func_class.$func_id;
312
313 }
314 if(file_exists(DIR_ROOT.'/question/'.strtolower(@$field[2]).'.q.php')){
315     include DIR_ROOT.'/question/'.strtolower(@$field[2]).'.q.php';
316 }
317
318
319 if ($nolable){
320     $isdiv = (@$func_div)? "</div>" : "" ;
321     return $file;
322 }else{
323     $isdiv = (@$func_div)? "</div>" : "" ;
324     return " <div><div class=\"table\"> $field[0]</div>
<div class=\"field\"> " .@$func_div.$file.$isdiv."</div></div><div
style=\"clear:both\"></div>\n" ;
325 }
326 }
327
328 /**
329  * Set the AutoRefresh function command
330  *
331  * @todo should be removed for JQuery ready({})
332  * @param String $command Javascript code or function
333  * @return void
334  */
335 function setAutoRefresh($command){
336     $this-> autoRefresh = " onChange=\" " .trim($command). "\" ;
337     $this-> autoRefreshcheckboxs = "
onClick=\" " .trim($command). "\" ;
338     $this-> autoRefreshInputEnter = "
onkeyup=\"entercontents(' " .trim($command). "', event);\" " ;
339 }
340
341 /**
342  * Remove the AutoRefresh function command
343  *
344  * @todo should be removed for JQuery ready({})
345  * @return void
346  */
347 function unsetAutoRefresh(){
348     $this-> autoRefresh = NULL;
349     $this-> autoRefreshcheckboxs = NULL;
350     $this-> autoRefreshInputEnter = NULL;
351 }
352 }
353 ?>

```

# File Source for table.class.php

Documentation for this file is available at [table.class.php](http://www.phpdoc.org/projects/phpdocu/table.class.php)

```
1  <?php
2  /**
3   * Template Class,
4   * <br />
5   * This class is used to take a input to generate templates<br />
6   *
7   * Example:<br />
8   * <br />
9   * $template = new template('template/index.html');<br />
10  * <br />
11  * $template->assign('var1', 'Employment list');<br />
12  * $template->assignArray(array{'var2'=>'John', 'var3'=>'mary',
13  * 'var4'=>'<strong>frank</strong>'});<br />
14  * <br />
15  * $ArrayVars[0]['name'] = 'john';<br />
16  * $ArrayVars[0]['age'] = '14';<br />
17  * $ArrayVars[1]['name'] = 'mary';<br />
18  * $ArrayVars[1]['age'] = '42';<br />
19  * $ArrayVars[2]['name'] = 'frank';<br />
20  * $ArrayVars[2]['age'] = '98';<br />
21  * <br />
22  * $template->assignRepeat('agelist', $ArrayVars);<br />
23  * $template->replace('../', '../..');<br />
24  * <br />
25  * $template->display();<br />
26  * <br />
27  * @author Jason Stewart <jason@lexxcom.com.au>
28  * @version 1.0
29  * @package PeopleScope
30  * @subpackage Base
31  */
32
33
34  class table{
35
36      private $identifier;
37      private $headerArray = array();
38      private $filterArray = array();
39      private $removeColumnArray = array();
40      private $columnsWidth = array();
41      private $id;
42      private $name = 'list';
43      private $basePage = NULL;
44      private $row_class_name;
45      private $row_class_field_name;
46      private $link_action = 'show';
47      private $link_field;
48      private $footer;
49      private $nolink = false;
50      private $rowsOnly = false;
51      private $SEOurl=SEO_LINK;
52
53
54      function __construct($id = NULL){
55          $this-> id = (!empty($id))? $id : NULL;
56          //by default get the current file name
57          $break = explode('/', $_SERVER['PHP_SELF']);
58          $pfile = $break[count($break) - 1];
59          //strip off the .php and store it
60          $this-> basePage = str_ireplace('.php', '', $pfile);
61          $this-> identifier_link_page = $this-> basePage;
62          $this-> SEOurl = (SEO_LINK===true) ? true : false;
63      }
64
65      function __destruct(){
66
```

```

67     }
68
69     public function setHeader($headerArray){
70         $this-> headerArray=$headerArray;
71     }
72
73     public function setRowsOnly(){
74         $this-> rowsOnly = true;
75     }
76
77     public function setFooter($field){
78         $this-> footer=$field;
79     }
80
81     public function setFilter($filterArray){
82         $this-> filterArray=$filterArray;
83     }
84
85     public function removeColumn($columnArray){
86         $this-> removeColumnArray=$columnArray;
87     }
88
89     public function setColumnsWidth($columnWidthArray){
90         $this-> columnsWidth=$columnWidthArray;
91     }
92
93     public function setColumnsClass($columnClassArray){
94         $this-> columnsWidth=$columnClassArray;
95     }
96
97     public function setIdentifier($field, $no_link=false){
98         $this-> identifier=$field;
99
100         //if not link is true, then only generate table with the identifier as the ID=''
101         if($no_link){
102             $this-> nolink=true;
103         }
104     }
105
106     public function setLinkAction($action){
107         $this-> link_action = $action;
108     }
109
110     public function setLinkField($action){
111         $this-> link_field = $action;
112     }
113
114     public function setIdentifierPage($page){
115         if($this-> SEUrl){
116             $page = (str_ireplace('/', '', str_ireplace('.php', '', $basepage)));
117         }
118         $this-> identifier_link_page = $page;
119     }
120
121     public function setTableName($name){
122         $this-> name = $name;
123     }
124
125     public function setBasePage($basepage){
126         $this-> basePage = (str_ireplace('/', '', str_ireplace('.php', '', $basepage)));
127     }
128
129     public function setPrimaryId($id){
130         $this-> id = (!empty($id))? $id : NULL;
131     }
132
133     /**
134      * Set the Class name for a Row in the table
135      * @param $name
136      * @return void
137      */
138     public function setRowClassName($name){
139         $this-> row_class_name = (!empty($name))? $name : NULL;
140     }
141
142     public function setRowClassFieldName($name){
143         $this-> row_class_field_name = (!empty($name))? $name : NULL;
144     }
145
146     public function generateDisplayTable($content){

```

```

147 //set vars
148 $id = '';
149 $column = '';
150 $filter = '';
151 $columnVal = '';
152 $filterVal = '';
153 //copy of main array for filter
154 $content2 = $content;
155 $str = '';
156 $td = '';
157 $link = '';
158 $CSS_id='';
159 $cwidth = '';
160
161 //return if nothing to build
162 /*if (count($content) == 0 || $content == false){
163     return "None";
164 }*/
165 //cycle echo through the records
166 if($content > 0){
167     foreach($content as $columnKey=> $val)
168     {
169
170         // check if we had made a list for the header
171         if(!$column){
172             //cycle thought each field list, this being the first need to also
173             create header
174             $countColumn = 0;
175             foreach($val AS $key=> $value){
176
177                 //if(isset($this->row_class_field_name)){
178                 if($key == $this-> row_class_field_name){
179                     $className = $value." " . $this->
180                     row_class_name;
181                 }else{
182                     $className = $this-> row_class_name;
183                 }
184             }
185
186             //is there a need to reset the header names
187             if (is_array($this-> headerArray)){
188                 //if current header is in headerArray change the display
189                 header name
190                 if(array_key_exists($key, $this-> headerArray)){
191                     $columnVal = $this-> getOrderType($this->
192                     headerArray[$key], $key);
193                 }else{
194                     $columnVal = $this-> getOrderType($key, $key); //just
195                     add header, no change
196                 }
197             }else{
198                 $columnVal = $this-> getOrderType($key, $key); //just
199                 add header, no change
200             }
201
202             //if the identifier used for the linking to a page, e.g primary
203             id
204             if($key == $this-> identifier){
205                 $id = $val[$key];
206                 $page = (isset($this-> identifier_link_page))? $this->
207                 identifier_link_page : $_SERVER['PHP_SELF'];
208                 if($this-> nolink){
209                     $link = "id=" . $id;
210                 }else{
211                     if(isset($this-> link_field)){
212                         $action = $val[$this-> link_field];
213                     }elseif(isset($this-> link_action)){
214                         $action = $this-> link_action;
215                     }else{
216                         $action = "show" ;
217                     }
218                     if($this-> SEOurl){
219                         $link = $page . "/" . $action . "/" . $id . "'\"
220                         "onclick=\"location.href='\" . $id;
221                     }else{
222                         $action = "action=" . $action;
223                         $link = $page . "?" . $action . "&id=" . $id . "'\"&q
224

```

```

215 uot; id="      . $id;
216                                     }
217                                     // $link =
"onclick=\"location.href='\" . $page . \"?\" . $action . \"&id=\" . $id . \"'\&q
218 uot; id=" . $id;
219                                     }
220                                     }
221                                     if(!in_array($key, $this-> removeColumnArray)){
222                                     if(!empty($this-> columnsWidth[$countColumn])){
223                                     $cWidth = ' width=" ' . $this->
> columnsWidth[$countColumn] . \"'\" ;
224                                     }
225                                     $column
226                                     <th $cWidth> \" . $columnVal . </th>\" ; // append to header
227                                     $td .= $this-> buildTd($key, $value);
228                                     }
229                                     // is there a need to a filter row
230                                     if (is_array($this-> filterArray) || !$this-> rowsOnly){
231                                     //if current header is in filterArray change then add the
filter to the filter row
232                                     if(array_key_exists($key, $this-> filterArray)){
233                                     $filterVal = $this-> getFilterType($this->
> filterArray[$key], $key, $content2); // format filter type
234                                     }
235                                     if(!in_array($key, $this-> removeColumnArray)){
236                                     if($countColumn == 0){
237                                     $primaryid = (!empty($this-> id)) ? <input
type='hidden' name='id' value=" \" . $this-> id . \">\" : \"\";
238                                     $filter .= <td
NOWRAP>\" . $primaryid . $filterVal . </td>\" ; // append then Field to filter Row
239                                     }else{
240                                     $filter .= <td
NOWRAP>\" . $filterVal . </td>\" ; // append then Field to filter Row
241                                     }
242                                     $countColumn++;
243                                     }
244                                     }
245                                     }
246                                     }else{
247                                     //cycle thought each field in row
248                                     foreach($val AS $key=> $value){
249                                     if(!in_array($key, $this-> removeColumnArray)){
250                                     $td .= $this-> buildTd($key, $value);
251                                     }
252                                     }
253                                     //if(isset($this->row_class_field_name)){
254                                     if($key == $this-> row_class_field_name){
255                                     $className = $value . \" \" . $this->
> row_class_name;
256                                     }else{
257                                     $className = $this-> row_class_name;
258                                     }
259                                     }
260                                     }
261                                     }
262                                     }
263                                     //if the identifier used for the linking to a page, e.g primary
id
264                                     if($key == $this-> identifier){
265                                     $id = $val[$key];
266                                     $page = (isset($this-> identifier_link_page)) ? $this->
> identifier_link_page : $_SERVER['PHP_SELF'] ;
267                                     if($this-> nolink){
268                                     $link = \"id=" . $id;
269                                     }else{
270                                     if(isset($this-> link_field)){
271                                     $action = \"action=" . $val[$this->
> link_field];
272                                     }elseif(isset($this-> link_action)){
273                                     $action = \"action=" . $this-> link_action;
274                                     }else{
275                                     $action = \"action=show\" ;
276                                     }
277                                     }
278                                     $link =
\"onclick=\"location.href=" . $page . \"?\" . $action . \"&id=" . $id . \"'\&q
279 uot; id=" . $id;
280                                     }

```



```

280     }
281     }
282     }
283     $str .= "<tr " . $link . "
class=\" " . $className . " row\">" . $std . "</tr>\n" ; // wrap in table
row and append
284         $std = ' ';
285     }
286 }
287 $footer = (isset($this-> footer)) ? "<TFOOT>" . $this->
> footer . "</TFOOT>" : ' ';
288 unset($this-> footer);
289
290 if($this-> rowsOnly){
291     $table = $str;
292 }else{
293     $table = "<table class=\"table_lists\" cellpadding=\"0\"
cellspacing=\"0\" id=\" " . $this-> name . "\" page=\" " . $this->
> basePage . "\">\n\t $footer<tr
class=\"header\"> " . $column . "<tr class=\"filter
noprnt\">" . $filter . "</tr>\n\t" . $str . "</table>" ; //build
table
294     }
295
296     return $table;
297 }
298
299 private function getFilterType($type, $key, $content = NULL){
300     switch($type){
301         case 'TEXT' :
302             $field = "<input type=\"text\"
name=\"flt_\" . $key . \"\" />" ; break;
303         case 'COMPILED' :
304         case 'COMPILED' :
305             //lets get all the values for the content list
306             foreach($content AS $contentkey => $contentVal){
307                 $contentArray[] = $contentVal[$key];
308             }
309             //compact and sort
310             $contentOrder = array_unique($contentArray);
311             sort($contentOrder);
312             //make the options list
313             $optionList = "<option value=\"NULL\"></option>" ;
314             foreach($contentOrder AS $val){
315                 $optionList .= "<option
value=\" " . $val . "\"> " . $val . "</option>" ;
316             }
317             $field = "<SELECT
name=\"flt_sel_\" . $key . \"\"> " . $optionList . "</SELECT>" ; break;
318         case 'VALUE' :
319             $field = "<SELECT name=\"dir_\" . $key . \"\" >
<option value=\"eq\" selected>=</option>
<option value=\"greater\">>=</option>
<option value=\"less\"><=</option>
</SELECT>
<input type=\"text\" name=\"flt_\"
size=\"2\" />" . $key . "\"
320             ; break;
321         case 'MONEY' :
322             $field = "<SELECT name=\"dir_\" . $key . \"\" >
<option value=\"eq\" selected>=</option>
<option value=\"greater\">>=</option>
<option value=\"less\"><=</option>
</SELECT>
<input type=\"text\" name=\"flt_\"
size=\"2\" />" . $key . "\"
323             ; break;
324         default: $field = ' ';
325     }
326
327     return $field;
328 }
329
330 private function getOrderType($type, $key){
331
332     $field = "<div
id=\" " . $key . "\"> " . $type . "</div>" ;
333
334     return $field;
335 }
336
337
338
339
340
341
342
343
344
345

```

```

346 private function buildTd($key, $value){
347
348     if(array_key_exists($key, $this-> filterArray)){
349
350         $type = $this-> filterArray[$key];
351         switch($type){
352             case 'VALUE' : $td = "<td style=\"text-
align:center\>" . $value . "</td>" ; break; // append then Field to Row
353             case 'MONEY' : $td = "<td style=\"text-
align:center\>" . number_format($value) . "</td>" ; break; // append then Field
to Row
354             default : $td = "<td>" . $value . "</td>" ;
355         }
356     }else{
357         $value2 = str_replace(',', '', $value);
358         if(is_numeric($value2)){
359             $td = "<td style=\"text-
align:center\>" . $value . "</td>" ;
360         }else{
361             $td = "<td>" . $value . "</td>" ; // append then Field to
Row
362         }
363     }
364 }
365
366 return $td;
367 }
368
369 public function buildWhereArrayFromRequest($full_like=NULL){
370
371     //set vars
372     $filter = false;
373
374     foreach($_REQUEST AS $key=> $val){
375         if (substr($key, 0 ,4) == 'flt_'){
376
377             $field = str_replace(substr($key, 0 ,4), '', $key);
378
379             if(isset($_REQUEST['dir_' . $field])){
380
381                 switch(trim($_REQUEST['dir_' . $field])){
382                     case 'greater' : $direction = ' >= ' ; break;
383                     case 'less' : $direction = ' <= ' ; break;
384                     case 'eq' : $direction = ' = ' ; break;
385                     default : $direction = ' = ' ; break;
386                 }
387                 $filter[] = $field . $direction . $val;
388             }else{
389                 if (substr($field, 0 ,4) == 'sel_'){
390                     if ($val != 'NULL' && !empty($val)){
391                         $filter[] = str_replace(substr($field, 0 ,4), '', $field) . " =
" . $val . " " ;
392                     }
393                 }else{
394                     if ($val != 'NULL' && !empty($val)){
395                         if($full_like){
396                             $filter[] = $field . " LIKE '%" . $val . "%' " ;
397                         }else{
398                             $filter[] = $field . " LIKE '" . $val . "' " ;
399                         }
400                     }
401                 }
402             }
403         }
404     }
405
406     return $filter;
407 }
408 }

```

# File Source for tools.function.php

Documentation for this file is available at [tools.function.php](http://tools.function.php)

```
1  <?php
2  /**
3   * Tools Function,
4   * <br />
5   * At set of general tool to help with development<br />
6   *
7   *
8   * @author Jason Stewart <jason@lexxcom.com.au>
9   * @version 1.0
10  * @package PeopleScope
11  * @subpackage Base
12  */
13
14
15  /**
16   * Global style for error messages
17   * @global string $GLOBALS['style']
18   */
19  $GLOBALS['style'] = "style=\"font-family: Arial, Helvetica, sans-serif; font-size: 11px;
padding:10px; background-color: #ffcccc; width:100%; border:solid 1px #000\"";
20  /**
21   * Global trace debugging
22   * @global string $GLOBALS['trace']
23   */
24  $GLOBALS['trace']=array();
25
26  /**
27   * A debugging tool
28   *
29   * This will produce a error message to the screen displaying the value enter
30   * it will take in vars | arrays | object
31   * it will show the page | Line | function that the echo accured
32   * and can be turned off thought the constant TURN_ON_PP
33   *
34   * @param Mixed $var Value that will be echoed out
35   * @param bool $tracer true display to screen, false store in global $trace
36   * return void
37   */
38  Function pp($var, $tracer=false){
39      if (TURN_ON_PP){
40          global $style;
41          $info = debug_backtrace();
42          $line = trim(str_replace($_SERVER['DOCUMENT_ROOT'], '', $info[0]['file']))." -- Line
: " . $info[0]['line']. " -- Function : " . $info[1]['function'];
43
44          if(is_array($var) || is_object($var)){
45              $response = " <div $style><h2>Notification
message</h2><h3> " . $line."</h3>\n
```

```

63  * @see showTrace for output
64  * @param Mixed $var Value that will be echoed out
65  * @param Bool $newline If type strip tags from string
66  */
67
68  function trace($var, $newline=true) {
69      global $trace, $style;
70      $info = debug_backtrace();
71      if (!$newline && is_string($var)) $var = trim(strip_tags($var));
72      $line = trim(str_replace($_SERVER['DOCUMENT_ROOT'], '', $info[0]['file']))." -- Line :
73      ".$info[0]['line']. " -- Function : " .@$info[1]['function'];
74      $response =
75      "<span>" . $line . "\n<pre>" . print_r($var,true) . "</pre></span>";
76      ;
77      $trace[] = $response;
78  }
79
80  /**
81   * Output $GLOBAL['trace'] Trace
82   * @global Array $trace
83   * @return Void
84   */
85  function showTrace() {
86      global $trace;
87      echo "<pre>" . print_r($trace,1) . "</pre>" ;
88  }
89
90  /**
91   * Will return a html form with all the results of the $GLOBAL vars
92   * $_POST, $_GET, $_COOKIE, $_SESSION, $trace
93   * @return String html vars
94   */
95  function showVars(){
96      if(!DEBUG){
97          return;
98      }
99      $ret = '<style type="text/css">
100      .debug h4 { margin-top: 0;
101      margin-bottom:0;
102      }
103      .debug h4:before {content: "+: ";}
104      .debug {float:none;
105      font-size: 0.8em;
106      background:#fff;
107      color: #999999;
108      height: 1.2em;
109      width: 1em;
110      border: 1px solid #999999;
111      overflow: hidden;
112      position:absolute;
113      top:0px;
114      left:0px;
115      }
116      .debug:hover { height: auto;
117      width: auto;
118      overflow: auto;
119      width:100%;}
120      </style>' ;
121      global $trace;
122      $ret .= '<div class="debug">' ;
123      $ret .= formatArray($_POST,'_POST');
124      $ret .= formatArray($_GET,'_GET');
125      $ret .= formatArray($_COOKIE,'_COOKIE');
126      $ret .= formatArray(@$_SESSION,'_SESSION');
127      $ret .= formatArray($trace,'TRACE');
128      $ret .= '</div>' ;
129
130      return $ret;
131  }
132
133  /**
134   * Will input an associate array and output in a readable format
135   * @param Array $array Array to read
136   * @param String $name name you want to give the array
137   */
138  function formatArray($array, $name){
139      $line='';

```

```

141     $line .= "<h4>" . $name . "</h4><blockquote><br />" ;
142     if (isset($array)){
143         foreach($array AS $key=> $value){
144             $line .= "[" . $key . "]=&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~";
145         }
146     }
147     $line .= "</blockquote>" ;
148     return $line;
149 }
150
151 /**
152  * Will take in a String and a start and end delimiter and will return the content between the
153  * delimiter
154  * @param String $string String to parse
155  * @param String $beg_tag start delimiter
156  * @param String $close_tag end delimiter
157  * @param Bool $remove_tag true return content without delimiter
158  * @return Array all content found between delimiters
159 */
160 function parseArray($string, $beg_tag, $close_tag, $remove_tag=true){
161     $on = preg_match_all("($beg_tag(.*)$close_tag)siU", $string, $matching_data);
162
163     if($remove_tag){
164         return $matching_data[1][0];
165     }
166
167     return $matching_data[0][0];
168 }
169
170 /**
171  * Will take in a list of associated array names and remove
172  * from that array via reference
173  * @param String $elements Comma seperated list of names
174  * @param Array $array The array to remove them from via reference
175 */
176 function stripElements($elements, & $array){
177     $elementsArray = explode(',', $elements);
178
179     foreach($elementsArray AS $value){
180         unset($array[$value]);
181     }
182 }
183
184 /**
185  * This will take in a name for the field types and a date that you wish and format
186  * an option list for select boxes, and return them in an associated array for year, month and day
187  * @param String $name Name used for selection box
188  * @param String $date Date That should be selected
189  * @param Integer $yearRange How many years should be displayed
190  * @param Integer $startyear What year should the list start at
191  * @return an Array of day,month,Year holding the required option list for select box
192 */
193 function createDateField($date, $yearRange = 10, $startyear = NULL){
194
195     if (!isset($startyear)){
196         $startyear = date('Y');
197     }
198
199     //lets do some cleaning up on the date format to make it easy to work with.
200     $date = trim($date);
201     $date = str_replace(array('\','/'), '-', $date);
202     $date = str_replace(' ', '', $date);
203
204     list($day, $month, $year) = explode('-', $date);
205
206     $months = array("", "January", "February", "March",
207 "April", "May", "June", "July", "August",
208 "September", "October", "November", "December");
209
210     for($i = 1; $i < count($months) ; $i++){
211         $selectmonth = ($i == $month)? 'selected="selected"' : '';
212         @$dateRet['month'] .= "<option value=\"$i\" $selectmonth> " . $months[$i] . "</option>\n";
213     }

```

```

215
216     for($j = 1; $j < 31; $j++){
217         $selectday = ($j == $day)? ' selected="selected"' : '';
218         @$dateRet['day'] .= " <option
value=\" $j\" $selectday> $j</option>\n " ;
219     }
220
221     for($k = $startyear; $k < ( $startyear+$yearRange) ; $k++){
222         $selectyear = ($k == $year)? ' selected="selected"' : '';
223         @$dateRet['year'] .= " <option
value=\" $k\" $selectyear> $k</option>\n " ;
224     }
225
226     return $dateRet;
227 }
228
229 /**
230  * Will take a Au date from jQuery Datepicker and convert to a database date format
231  *
232  * @link http://docs.jquery.com/UI/Datepicker
233  * @link http://docs.jquery.com/UI/Datepicker/formatDate
234  * @param String $date database date format
235  */
236 function formatDateUI($date){
237     //exit(print_r(debug_backtrace()));
238     list($day, $month, $year) = explode('/', $date);
239     $year = (strlen($year) == 2)? '20'.$year: $year;
240     return $year.'-'. $month.'-'. $day.' 00:00:00' ;
241 }
242
243
244 /**
245  * This will take in an Array or if no array give will default to the $_REQUEST
246  * global looking for associated name type and remove the elements and replace the with a $name
247  * field with a value of the compiled date,
248  * if none found then will put in todays date
249  * associated name types :<br />
250  * $name.'__day'<br />
251  * $name.'__month'<br />
252  * $name.'__year'<br />
253  * <br />
254  * $name = $name.'__year-'$name.'__month-'$name.'__day 00::00::00' <br />
255  * <br />
256  * @param String $name the name of the fields we are looking for
257  * @param Array &$listArray alternate array to look at, $_REQUEST by default
258  * @return String database date format
259  */
260 function formatDateResponse($name, & $listArray=NULL){
261     if (!$listArray){
262         $listArray = $_REQUEST;
263     }
264
265     if(isset($listArray[$name.'__day'])){
266         $nameDay = $listArray[$name.'__day'];
267         unset($listArray[$name.'__day']);
268     }else{
269         $nameDay = date('j');
270     }
271
272     if(isset($listArray[$name.'__month'])){
273         $nameMonth = $listArray[$name.'__month'];
274         unset($listArray[$name.'__month']);
275     }else{
276         $nameMonth = date('n');
277     }
278
279     if(isset($listArray[$name.'__year'])){
280         $nameYear = $listArray[$name.'__year'];
281         unset($listArray[$name.'__year']);
282     }else{
283         $nameYear = date('Y');
284     }
285
286     $listArray[$name] = ($nameYear.'-'. $nameMonth.'-'. $nameDay.' 00:00:00' );
287
288     return $listArray[$name];
289 }
290
291 /**
292  * Will convert the output from a JQuery UI date field format

```

```

292 * format must be in Au date format
293 * @link http://docs.jquery.com/UI/Datepicker
294 * @link http://docs.jquery.com/UI/Datepicker/formatDate
295 *
296 * @todo remember how this works :S
297 * @param String $start start date
298 * @param String $end end date
299 * @param Array &$ARRAY returns by reference
300 */
301 function convertUIdate($start, $end, & $ARRAY){
302     $startdate = (empty($start))? '' : $ARRAY[$start];
303     $enddate = (empty($end))? '' : $ARRAY[$end];
304
305     if(empty($startdate) && empty($enddate)){
306         return;
307     }
308
309     if(!empty($startdate)){
310         list($sd,$sm,$sy) = explode('/', $startdate);
311         $stimestamp = mktime(0,0,0,$sm,$sd,$sy);
312         if(empty($enddate)){
313             $ARRAY[$start] = date('Y-m-d 00:00:00', $stimestamp);
314             return;
315         }
316     }
317
318     if(!empty($enddate)){
319         list($ed,$em,$ey) = explode('/', $enddate);
320         $setimestamp = mktime(0,0,0,$em,$ed,$ey);
321         if(empty($startdate)){
322             $ARRAY[$end] = date('Y-m-d 00:00:00', $setimestamp);
323             return;
324         }
325     }
326
327     if ($stimestamp < $setimestamp){
328         $ARRAY[$start] = date('Y-m-d 00:00:00', $stimestamp);
329         $ARRAY[$end] = date('Y-m-d 00:00:00', $setimestamp);
330     }else{
331         $ARRAY[$end] = date('Y-m-d 00:00:00', $stimestamp);
332         $ARRAY[$start] = date('Y-m-d 00:00:00', $setimestamp);
333     }
334 }
335
336 /**
337 * converts a database date value and converts to Au date format
338 * 2001-04-02 = 02/04/2001
339 * @param String $dbdate database date format yyyy-mm-dd
340 * @return String Date dd/mm/yyyy
341 */
342 function databaseToUI($dbdate){
343     list($dbdate) = explode(' ', $dbdate);
344
345     list($year, $month, $day) = explode('-', $dbdate);
346
347     return $day."/". $month."/". $year;
348 }
349
350 /**
351 * return file extension based on Mime type for common doc types
352 * doc, docx, pdf
353 *
354 * @param String $type Mime type
355 * @return String Document extension
356 */
357 function fileExt($type){
358     $ext = false;
359     switch($type){
360         case 'application/msword' : $ext = 'doc'; break;
361         case 'application/pdf' : $ext = 'pdf'; break;
362         case 'application/vnd.openxmlformats-officedocument.wordprocessingml.document' : $ext =
363 'docx'; break;
364     }
365     return $ext;
366 }
367
368 /**
369 * Will take in a percent amount and return values in mins
370 * eg. 50% = 30mins

```

```

371  * @param Integer $percent
372  * @return Integer Minutes
373  */
374  function convertDecimalToMinutes($percent){
375      list($hours, $minutes)= explode('.', number_format($percent,2));
376      $minutes = ceil(($minutes*60)/100);
377      $in_minutes=($hours*60)+$minutes;
378      return $in_minutes;
379  }
380
381  /**
382   * Will take in a amount in minutes and return the value in the amount of hours : minute
383   *
384   * 124 minutes = 2:04
385   *
386   * @param Integer $Minutes
387   * @return String hh:mm
388   */
389  function convertMinutes2Hours($Minutes)
390  {
391      if ($Minutes < 0)
392      {
393          $Min = Abs($Minutes);
394      }
395      else
396      {
397          $Min = $Minutes;
398      }
399      $iHours = Floor($Min / 60);
400      $Minutes = ($Min - ($iHours * 60)) / 100;
401      $tHours = $iHours + $Minutes;
402      if ($Minutes < 0)
403      {
404          $tHours = $tHours * (-1);
405      }
406      $aHours = explode(".", $tHours);
407      $iHours = $aHours[0];
408      if (empty($aHours[1]))
409      {
410          $aHours[1] = "00" ;
411      }
412      $Minutes = $aHours[1];
413      if (strlen($Minutes) < 2)
414      {
415          $Minutes = $Minutes ."0" ;
416      }
417      $tHours = $iHours .":" . $Minutes;
418      return $tHours;
419  }
420
421  function Box($content, $header, $discription){
422      $html = '<div class="tab">
423          <div id="tab-header">
424              <h1>' . $header . '</h1>
425          </div>
426          <div id="tab-text">
427              ' . $discription . '
428          </div>
429          <div id="tab-body">
430              ' . $content . '<br class="clear">
431          </div>
432          <div id="tab-foot"></div>
433      </div>' ;
434      return $html;
435  }
436
437  function warning($content){
438      $html = '<div class="warning">
439          ' . $content . '
440      </div>' ;
441      return $html;
442  }
443  }
444  }
445  ?>

```



# 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 [advertisement::getSelectListOfCategory\(\)](#)

- move this to its own class for all classes to use

In [advertisement::getSelectListOfEmploymentType\(\)](#)

- move this to its own class for all classes to use

In [advertisement::getSelectListOfStates\(\)](#)

- move this to its own class for all classes to use

In [advertisement::getSelectListOfTemplate\(\)](#)

- move this as a function in advertTemplate class
- move this to its own class for all classes to use

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

- should be removed for JQuery ready({})

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

- confirm this is correct

In [advertisement::templateAdvertisementLayout\(\)](#)

- find out what \$inputArray is used for

In [advertTemplate::templateAdvertTemplateLayout\(\)](#)

- find out what \$inputArray is used for

In [category::templateCategoryLayout\(\)](#)

- find out what \$inputArray is used for

In [employmenttype::templateEmploymenttypeLayout\(\)](#)

- find out what \$inputArray is used for

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

- should be removed for JQuery ready({})

# Index

## A

<a href="#">advertTemplate::\$validation_error</a>	36
<i>Array use to store any error found during Validation function</i>	
<a href="#">advertTemplate::\$template</a>	36
<i>Template class object</i>	
<a href="#">advertTemplate::create()</a>	36
<i>This method will take an array and insert it in the database</i>	
<a href="#">advertTemplate::createAdvertTemplateDetails()</a>	37
<i>This method will provide a page to the to add a single row AdvertTemplate to the template table</i>	
<a href="#">advertTemplate::editAdvertTemplateDetails()</a>	38
<i>Show the details ready to edit of a single AdvertTemplate from the template</i>	
<a href="#">advertTemplate::deleteAdvertTemplateDetails()</a>	38
<i>Set a row to be marked as deleted</i>	
<a href="#">advertTemplate::\$table</a>	35
<i>Table class object</i>	
<a href="#">advertTemplate::\$fields_validation_type</a>	35
<i>Array of feilds and there types that are check when validating</i>	
<a href="#">advertTemplate::\$db</a>	34
<i>Database class object</i>	
<a href="#">advertTemplate</a>	33
<i>AdvertTemplate Class</i>	
<i>&lt;pre&gt;</i>	
<i>This class is based on the table template</i>	
<a href="#">advertTemplate::\$db_connect</a>	34
<i>Connect to PDO object through database class</i>	
<a href="#">advertTemplate::\$fields</a>	34
<i>Array of field used in the database if not in this list is dropped from insert or update</i>	
<a href="#">advertTemplate::\$fields_required</a>	35
<i>Array of feilds require information when validating</i>	
<a href="#">advertTemplate::getAdvertTemplateList()</a>	39
<i>Show list of information corresponding the to this class</i>	
<a href="#">advertTemplate::getSelectListOfCategory()</a>	39
<i>This Method will either the catagory_name feild or a select box of catagory_name fields</i>	
<a href="#">advertTemplate::update()</a>	44
<i>This method will take an array and update a row in the database</i>	
<a href="#">advertTemplate::templateAdvertTemplateLayout()</a>	43
<i>This method assigns the associate array values to the template</i>	
<a href="#">advertTemplate::updateAdvertTemplateDetails()</a>	45
<i>update the information in a single AdvertTemplate from the template</i>	
<a href="#">advertTemplate::Validate()</a>	45
<i>This medthod is used to validate inputs from form information</i>	
<a href="#">advertTemplate.class.php</a>	132
<i>Source code</i>	
<a href="#">advertisement.class.php</a>	117

Source code	
<a href="#">advertTemplate::showAdvertTemplateDetails()</a>	43
Show details of a single AdvertTemplate from the template	
<a href="#">advertTemplate::saveAdvertTemplateDetails()</a>	42
save the information in a single AdvertTemplate to the template table	
<a href="#">advertTemplate::getSelectListOfStates()</a>	40
This Method will either the name feild or a select box of state fields	
<a href="#">advertTemplate::getSelectListOfEmploymentType()</a>	40
This Method will either the employmenttype feild or a select box of employmenttype fields	
<a href="#">advertTemplate::lists()</a>	41
Show will pull a list from the corresponding AdvertTemplate template	
<a href="#">advertTemplate::read()</a>	41
This method will return information as row	
<a href="#">advertTemplate::remove()</a>	42
This method will update a row and make the recored as deleted	
<a href="#">advertisement::Validate()</a>	33
This medthod is used to validate inputs from form information	
<a href="#">advertisement::updateAdvertisementDetails()</a>	32
update the information in a single Advertisement from the advertisement	
<a href="#">advertisement::\$template</a>	22
Template class object	
<a href="#">advertisement::\$table</a>	22
Table class object	
<a href="#">advertisement::\$validation_error</a>	22
Array use to store any error found during Validation function	
<a href="#">advertisement::create()</a>	23
This method will take an array and insert it in the database	
<a href="#">advertisement::deleteAdvertisementDetails()</a>	24
Set a row to be marked as deleted	
<a href="#">advertisement::createAdvertisementDetails()</a>	24
This method will provide a page to the to add a single row Advertisement to the advertisement table	
<a href="#">advertisement::\$fields_validation_type</a>	21
Array of feilds and there types that are check when validating	
<a href="#">advertisement::\$fields_required</a>	21
Array of feilds require information when validating	
<a href="#">advertisement</a>	20
Advertisement Class	
<pre>&lt;pre&gt;</pre>	
This class is based on the table advertisement	
<a href="#">advertTemplate.class.php</a>	14
<a href="#">advertisement::\$db</a>	20
Database class object	
<a href="#">advertisement::\$db_connect</a>	20
Connect to PDO object through database class	
<a href="#">advertisement::\$fields</a>	21
Array of field used in the database if not in this list is dropped from insert or update	
<a href="#">advertisement::editAdvertisementDetails()</a>	25
Show the details ready to edit of a single Advertisement from the advertisement	
<a href="#">advertisement::getAdvertisementList()</a>	25
Show list of information corresponding the to this class	
<a href="#">advertisement::saveAdvertisementDetails()</a>	30
save the information in a single Advertisement to the advertisement table	

<a href="#">advertisement::remove()</a>	29
<i>This method will update a row and make the record as deleted</i>	
<a href="#">advertisement::showAdvertisementDetails()</a>	30
<i>Show details of a single Advertisement from the advertisement</i>	
<a href="#">advertisement::templateAdvertisementLayout()</a>	31
<i>This method assigns the associate array values to the template</i>	
<a href="#">advertisement::update()</a>	31
<i>This method will take an array and update a row in the database</i>	
<a href="#">advertisement::read()</a>	29
<i>This method will return information as row</i>	
<a href="#">advertisement::lists()</a>	28
<i>Show will pull a list from the corresponding Advertisement advertisement</i>	
<a href="#">advertisement::getSelectListOfEmploymentType()</a>	26
<i>This Method will either the employmenttype feild or a select box of employmenttype fields</i>	
<a href="#">advertisement::getSelectListOfCategory()</a>	26
<i>This Method will either the catagory_name feild or a select box of catagory_name fields</i>	
<a href="#">advertisement::getSelectListOfStates()</a>	27
<i>This Method will either the name feild or a select box of state fields</i>	
<a href="#">advertisement::getSelectListOfTemplate()</a>	27
<i>This Method will either the employmenttype feild or a select box of employmenttype fields</i>	
<a href="#">advertisement::getUserById()</a>	28
<i>Will get name of the user based on id</i>	
<a href="#">advertisement.class.php</a>	13

## B

<a href="#">Box()</a>	74
-----------------------	----

## C

<a href="#">convertDecimalToMinutes()</a>	74
<i>Will take in a percent amount and return values in mins</i>	
<a href="#">convertMinutes2Hours()</a>	74
<i>Will take in a amount in minutes and return the value in the amount of hours : minute</i>	
<a href="#">convertUIdate()</a>	75
<i>Will convert the output from a JQuery UI date field format</i>	
<a href="#">createDateField()</a>	75
<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>	
<a href="#">constructor employmenttype::__construct()</a>	60
<i>Contructor for this method</i>	
<a href="#">category::Validate()</a>	57
<i>This medthod is used to validate inputs from form information</i>	
<a href="#">category::showCategoryDetails()</a>	54
<i>Show details of a single Category from the category</i>	
<a href="#">category::templateCategoryLayout()</a>	55
<i>This method assigns the associate array values to the template</i>	
<a href="#">category::update()</a>	55
<i>This method will take an array and update a row in the database</i>	
<a href="#">category::updateCategoryDetails()</a>	56
<i>update the information in a single Category from the category</i>	

<a href="#">CustomException</a>	79
<i>Generates a custom exception</i>	
<a href="#">CustomException::logError()</a>	79
<i>Generate a formatted exception error</i>	
<a href="#">constructor table::__construct()</a>	95
<a href="#">constructor template::__construct()</a>	102
<i>Constructor for template class</i>	
<a href="#">constructor template::template()</a>	103
<i>Constructor php4 for template class</i>	
<a href="#">category.class.php</a>	154
<i>Source code</i>	
<a href="#">constructor form::form()</a>	89
<i>Constructor php4</i>	
<a href="#">constructor form::__construct()</a>	89
<i>Constructor</i>	
<a href="#">CustomException::messageError()</a>	79
<a href="#">CustomException::queryError()</a>	79
<i>Will append the query being used to the begining of a logError output</i>	
<a href="#">constructor db::__construct()</a>	80
<i>constructor to initiate database conection</i>	
<a href="#">constructor email::email()</a>	84
<i>Constructor for Email php4</i>	
<a href="#">category::saveCategoryDetails()</a>	53
<i>save the information in a single Category to the category table</i>	
<a href="#">category::remove()</a>	53
<i>This method will update a row and make the recored as deleted</i>	
<a href="#">category::\$db connect</a>	47
<i>Connect to PDO object through database class</i>	
<a href="#">category::\$fields</a>	47
<i>Array of field used in the database if not in this list is dropped from insert or update</i>	
<a href="#">category::\$fields_required</a>	47
<i>Array of feilds require information when validating</i>	
<a href="#">category::\$fields_validation_type</a>	48
<i>Array of feilds and there types that are check when validating</i>	
<a href="#">category::\$db</a>	46
<i>Database class object</i>	
<a href="#">category</a>	46
<i>Category Class</i>	
<i>&lt;pre&gt;</i>	
<i>This class is based on the table category</i>	
<a href="#">constructor template::__construct()</a>	7
<a href="#">category.class.php</a>	18
<a href="#">constructor advertisement::__construct()</a>	22
<i>Constructor for this method</i>	
<a href="#">constructor advertTemplate::__construct()</a>	36
<i>Constructor for this method</i>	
<a href="#">category::\$table</a>	48
<i>Table class object</i>	
<a href="#">category::\$template</a>	48
<i>Template class object</i>	
<a href="#">category::editCategoryDetails()</a>	51
<i>Show the details ready to edit of a single Category from the category</i>	
<a href="#">category::getCategoryList()</a>	51

<i>Show list of information corresponding the to this class</i>	52
<a href="#">category::lists()</a>	52
<i>Show will pull a list from the corresponding Category category</i>	
<a href="#">category::read()</a>	53
<i>This method will return information as row</i>	
<a href="#">category::deleteCategoryDetails()</a>	50
<i>Set a row to be marked as deleted</i>	
<a href="#">category::createCategoryDetails()</a>	50
<i>This method will provide a page to the to add a single row Category to the category table</i>	
<a href="#">category::\$validation_error</a>	48
<i>Array use to store any error found during Validation function</i>	
<a href="#">constructor category::construct()</a>	49
<i>Contructor for this method</i>	
<a href="#">category::create()</a>	49
<i>This method will take an array and insert it in the database</i>	
<a href="#">constructor login::construct()</a>	5

## D

<a href="#">db::prepareToQuery()</a>	81
<i>Merge query template with array</i>	
<a href="#">db::insert()</a>	81
<i>Will instigate a INSERT query and return the inserts autocomplete Id;</i>	
<a href="#">db::query()</a>	82
<i>Will instigate a query and return a recordset;</i>	
<a href="#">db::select()</a>	82
<i>Will instigate a SELECT query and return and an array of responses</i>	
<a href="#">database.class.php</a>	176
<i>Source code</i>	
<a href="#">db::update()</a>	82
<i>Will instigate a UPDATE query and return true if no problems;</i>	
<a href="#">db::getDNSString()</a>	81
<i>returns current DSN string used to connect ot the server</i>	
<a href="#">db::delete()</a>	80
<i>Will instigate a DELETE query and return true if no problems;</i>	
<a href="#">db</a>	80
<i>This class is Database abstraction layer</i>	
<a href="#">databaseToUI()</a>	75
<i>converts a database date value and conversts to Au date format</i>	
<a href="#">db::\$dbh</a>	80
<a href="#">db::\$dsn</a>	80
<a href="#">db::\$lastQuery</a>	80
<a href="#">database.class.php</a>	70
<i>Database Class,</i>	
<i>&lt;br /&gt;</i>	
<i>This class is Database abstraction layer&lt;br /&gt;</i>	

## E

<a href="#">email::\$mime</a>	84
<i>Set Mime Type</i>	



<a href="#">email::\$mail</a>	83
<i>Mail Object</i>	
<a href="#">email::\$template</a>	84
<i>Template Object</i>	
<a href="#">email::append_file()</a>	84
<i>converts the TEXT version of the email to be sent to be appended to the email</i>	
<a href="#">email::append_html()</a>	84
<i>converts the HTML version of the email to be sent to be appended to the email</i>	
<a href="#">email::\$header</a>	83
<i>Mail Header information</i>	
<a href="#">email::\$crlf</a>	83
<i>Carrage return</i>	
<a href="#">employmenttype::updateEmploymenttypeDetails()</a>	67
<i>update the information in a single Employmenttype from the employmenttype</i>	
<a href="#">employmenttype::Validate()</a>	68
<i>This medthod is used to validate inputs from form information</i>	
<a href="#">email.class.php</a>	71
<i>Email Class,</i>	
<i>&lt;br /&gt;</i>	
<i>This class is used create and send email, can also record emails in a db, and can use the</i>	
<i>template class to format Html &lt;br /&gt;</i>	
<a href="#">email</a>	83
<i>This class is used create and send email&lt;br /&gt;</i>	
<a href="#">email::append_text()</a>	84
<i>converts the TEXT version of the email to be sent to be appended to the email</i>	
<a href="#">email::recordEmail()</a>	84
<i>Record email into the database</i>	
<a href="#">email::construct()</a>	86
<i>Constructor for Email php5</i>	
<a href="#">errorhandler.class.php</a>	146
<i>Source code</i>	
<a href="#">employmenttype.class.php</a>	165
<i>Source code</i>	
<a href="#">email.class.php</a>	180
<i>Source code</i>	
<a href="#">email::subject()</a>	86
<i>Set the Subject line fo the email</i>	
<a href="#">email::setTXTtemplate()</a>	86
<i>Will create and output from Form Class standardised format String</i>	
<i>To create a TEXT email in a reable format</i>	
<a href="#">email::send()</a>	85
<i>compile and Send email</i>	
<a href="#">email::sender()</a>	85
<i>Set the Sender info for header</i>	
<a href="#">email::setHeader()</a>	85
<i>Set all other Header information as required</i>	
<a href="#">email::setHTMLtemplate()</a>	85
<i>Create a HTML email using the base template class</i>	
<a href="#">employmenttype::update()</a>	66
<i>This method will take an array and update a row in the database</i>	
<a href="#">employmenttype::templateEmploymenttypeLayout()</a>	66
<i>This method assigns the associate array values to the template</i>	
<a href="#">employmenttype::\$fields</a>	58

<i>Array of field used in the database if not in this list is dropped from insert or update</i>	
<a href="#">employmenttype::\$fields_required</a>	59
<i>Array of feilds require information when validating</i>	
<a href="#">employmenttype::\$fields_validation_type</a>	59
<i>Array of feilds and there types that are check when validating</i>	
<a href="#">employmenttype::\$table</a>	59
<i>Table class object</i>	
<a href="#">employmenttype::\$db_connect</a>	58
<i>Connect to PDO object through database class</i>	
<a href="#">employmenttype::\$db</a>	58
<i>Database class object</i>	
<a href="#">exception_error_handler()</a>	15
<a href="#">exception_handler()</a>	15
<a href="#">employmenttype.class.php</a>	19
<a href="#">employmenttype</a>	57
<i>Employmenttype Class</i>	
<pre>&lt;pre&gt;</pre>	
<i>This class is based on the table employmenttype</i>	
<a href="#">employmenttype::\$template</a>	59
<i>Template class object</i>	
<a href="#">employmenttype::\$validation_error</a>	60
<i>Array use to store any error found during Validation function</i>	
<a href="#">employmenttype::read()</a>	64
<i>This method will return information as row</i>	
<a href="#">employmenttype::remove()</a>	64
<i>This method will update a row and make the recored as deleted</i>	
<a href="#">employmenttype::saveEmploymenttypeDetails()</a>	65
<i>save the information in a single Employmenttype to the employmenttype table</i>	
<a href="#">employmenttype::showEmploymenttypeDetails()</a>	65
<i>Show details of a single Employmenttype from the employmenttype</i>	
<a href="#">employmenttype::lists()</a>	63
<i>Show will pull a list from the corresponding Employmenttype employmenttype</i>	
<a href="#">employmenttype::getEmploymenttypeList()</a>	62
<i>Show list of information corresponding the to this class</i>	
<a href="#">employmenttype::create()</a>	60
<i>This method will take an array and insert it in the database</i>	
<a href="#">employmenttype::createEmploymenttypeDetails()</a>	61
<i>This method will provide a page to the to add a single row Employmenttype to the employmenttype table</i>	
<a href="#">employmenttype::deleteEmploymenttypeDetails()</a>	61
<i>Set a row to be marked as deleted</i>	
<a href="#">employmenttype::editEmploymenttypeDetails()</a>	62
<i>Show the details ready to edit of a single Employmenttype from the employmenttype</i>	
<a href="#">errorhandler.class.php</a>	15
<i>CustomException Class,</i>	
<pre>&lt;br /&gt;</pre>	
<i>Generates a custom exception&lt;br /&gt;</i>	

## F

<a href="#">form::formFooter()</a>	89
<i>Setup closing form tag</i>	

<a href="#">form::formHeader()</a>	89
Setup the form tag ready for the inputs	
<a href="#">form::draw()</a>	89
Will Generated the required input fields from the \$this->form information information can be broken up using the string '-----'	
<a href="#">form::\$formScript</a>	89
Java scripts reloaded to this class	
<a href="#">form::\$form</a>	88
Holds string value for form builder	
<a href="#">form::formScript()</a>	90
Set onSubmit Form scripts	
<a href="#">form::freeFormSubmit()</a>	90
Standards Submit Button	
<a href="#">form::unsetAutoRefresh()</a>	91
Remove the AutoRefresh function command	
<a href="#">form.class.php</a>	184
Source code	
<a href="#">form::SetFckConfigUrl()</a>	91
Set a alternate FCK config file path	
<a href="#">form::setAutoRefresh()</a>	90
Set the AutoRefresh function command	
<a href="#">form::inputfield()</a>	90
Takes in a standard single row from \$this->form information and generated and html form input string	
<a href="#">form::\$fck_width</a>	88
Width size of fck editor	
<a href="#">form::\$fck_height</a>	88
Height size of fck editor	
<a href="#">formatDateUI()</a>	76
Will take a Au date from jQuery Datepicker and convert to a database date format	
<a href="#">form</a>	87
This class is used to take a input String in a standardised format and convert to a Html Form	
<a href="#">formatDateResponce()</a>	76
This will take in an Array or if no array give will default to the \$_REQUEST	
<a href="#">formatArray()</a>	76
Will input an associate array and output in a readable format	
<a href="#">fileExt()</a>	76
return file extention based on Mime type for common doc types doc, docx, pdf	
<a href="#">form::\$autoRefresh</a>	87
JavaScript handle onChange for Form input types command	
<a href="#">form::\$autoRefreshcheckboxs</a>	87
JavaScript handle onClick for Form input types command	
<a href="#">form::\$fck_configUrl</a>	88
Url Tor find fck config file	
<a href="#">form::\$enctype</a>	88
Form enctype type	
<a href="#">form::\$db</a>	88
Database object	
<a href="#">form::\$autoRefreshInputEnter</a>	87
JavaScript handle onkeyup for Form input types command	
<a href="#">form.class.php</a>	72
Form Class,	

<br />

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

<br />

## G

<a href="#">global \$GLOBALS['trace']</a> . . . . .	78
Global trace debugging	
<a href="#">global \$GLOBALS['style']</a> . . . . .	78
Global style for error messages	

## L

<a href="#">login::\$template</a> . . . . .	5
<a href="#">login::checkUserLogin()</a> . . . . .	5
<a href="#">login::getHomePage()</a> . . . . .	5
<a href="#">login.class.php</a> . . . . .	115
Source code	
<a href="#">login::\$table</a> . . . . .	5
<a href="#">login::\$lastError</a> . . . . .	4
<a href="#">login</a> . . . . .	4
<a href="#">login::\$admin</a> . . . . .	4
<a href="#">login::\$error</a> . . . . .	4
<a href="#">login.class.php</a> . . . . .	3

## M

<a href="#">myErrorHandler()</a> . . . . .	15
--	----

## P

<a href="#">pp()</a> . . . . .	77
A debugging tool	
<a href="#">parseArray()</a> . . . . .	77
Will take in a String and a start and end delimiter and will return the content between the delimiter	

## S

<a href="#">stripElements()</a> . . . . .	78
Will take in a list of accociated array names and remove from that array via reference	
<a href="#">showVars()</a> . . . . .	77
Will return a html form with all the results of the \$GLOBAL vars	
<a href="#">showTrace()</a> . . . . .	77
Output \$GLOBAL['trace'] Trace	

# T

<a href="#">table::setIdentifierPage()</a>	99
<a href="#">table::setIdentifier()</a>	99
<a href="#">table::setHeader()</a>	99
<a href="#">table::setFooter()</a>	98
<a href="#">table::setLinkAction()</a>	99
<a href="#">table::setLinkField()</a>	100
<a href="#">table::setRowClassName()</a>	101
<i>Set the Class name for a Row in the table</i>	
<a href="#">table::setRowClassFieldName()</a>	100
<a href="#">table::setPrimaryId()</a>	100
<a href="#">table::setFilter()</a>	98
<a href="#">table::setColumnsWidth()</a>	98
<a href="#">table::generateDisplayTable()</a>	96
<a href="#">table::buildWhereArrayFromRequest()</a>	96
<a href="#">table::buildTd()</a>	95
<a href="#">table::getFilterType()</a>	96
<a href="#">table::getOrderType()</a>	97
<a href="#">table::setColumnsClass()</a>	98
<a href="#">table::setBasePage()</a>	97
<a href="#">table::removeColumn()</a>	97
<a href="#">table::setRowsOnly()</a>	101
<a href="#">table::setTableName()</a>	101
<a href="#">template::replace()</a>	106
<i>Will replace the string across the entire template</i>	
<a href="#">template::fetch()</a>	105
<i>Will return the full generated page template as a variable</i>	
<a href="#">template::display()</a>	105
<i>Will print out the full generated page template</i>	
<a href="#">template::BuildAssigned()</a>	105
<i>Build template with assigned values</i>	
<a href="#">template::set()</a>	106
<i>Set the current template path to a new template file</i>	
<a href="#">template.class.php</a>	112
<i>Source code</i>	
<a href="#">tools.function.php</a>	195
<i>Source code</i>	
<a href="#">table.class.php</a>	189
<i>Source code</i>	
<a href="#">template.old.class.php</a>	149
<i>Source code</i>	
<a href="#">template::assignRepeat()</a>	104
<i>Will assign to a repeating element from an assigned array &lt;br /&gt; based on array row and element name eg.</i>	
<a href="#">template::assignBlank()</a>	104
<i>Assign Blank is used if not using a template, example such as XML output</i>	
<a href="#">template::\$assigned</a>	102
<i>array of assigned values to a template</i>	
<a href="#">template</a>	102
<i>This class is used to take a input to generate templates</i>	
<a href="#">table::__destruct()</a>	101
<a href="#">template::\$repeatRegion</a>	102

<i>Array of Repeat region</i>	
<a href="#">template::\$replacer</a>	102
<i>Array of values that should be replaced in template</i>	
<a href="#">template::\$assignArray()</a>	103
<i>An Array of elements to be added to the template</i>	
<i>array{'jason'=&gt;'john', 'age'=&gt;'14'}</i>	
<a href="#">template::\$assign()</a>	103
<i>Used to assign a value element to a generated template</i>	
<a href="#">template::\$template_file</a>	102
<i>Location of template file</i>	
<a href="#">table::\$SEUrl</a>	95
<a href="#">table::\$row_class_name</a>	95
<a href="#">template::formatValue()</a>	9
<a href="#">template::formatBoolean()</a>	8
<a href="#">template::fetch()</a>	8
<a href="#">template::externalLink()</a>	8
<a href="#">template::getListTable()</a>	9
<a href="#">template::input()</a>	9
<a href="#">template::strip_tags()</a>	10
<a href="#">template::page()</a>	10
<a href="#">template::insert()</a>	10
<a href="#">template::display()</a>	8
<a href="#">template::content()</a>	7
<a href="#">template::\$db_connect</a>	6
<a href="#">template::\$db</a>	6
<a href="#">template</a>	5
<a href="#">template::\$filterArray</a>	6
<a href="#">template::\$headerArray</a>	6
<a href="#">template::assign()</a>	7
<a href="#">template::\$template</a>	7
<a href="#">template::\$layout</a>	6
<a href="#">template::__destruct()</a>	11
<a href="#">template.old.class.php</a>	17
<i>Template Class,</i>	
<i>&lt;br /&gt;</i>	
<i>This class is used to take a input to generate templates&lt;br /&gt;</i>	
<a href="#">table::\$link_field</a>	94
<a href="#">table::\$link_action</a>	94
<a href="#">table::\$identifier</a>	93
<a href="#">table::\$name</a>	94
<a href="#">table::\$nolink</a>	94
<a href="#">table::\$row_class_field_name</a>	95
<a href="#">table::\$rowsOnly</a>	95
<a href="#">table::\$removeColumnArray</a>	94
<a href="#">table::\$id</a>	93
<a href="#">table::\$headerArray</a>	93
<a href="#">trace()</a>	78
<i>Debuging tool that that will store the outcomes into a \$GLOBAL['trace'] var</i>	
<a href="#">tools.function.php</a>	74
<i>Tools Function,</i>	
<i>&lt;br /&gt;</i>	
<i>At set of general tool to help with development&lt;br /&gt;</i>	
<a href="#">table.class.php</a>	73

<a href="#">table</a>	91
<i>Template Class,</i>	
<i>&lt;br /&gt;</i>	
<i>This class is used to take a input to generate templates&lt;br /&gt;</i>	
<a href="#">table::\$basePage</a>	92
<a href="#">table::\$footer</a>	93
<a href="#">table::\$filterArray</a>	92
<a href="#">table::\$columnsWidth</a>	92
<a href="#">template.class.php</a>	2

## W

<a href="#">warning()</a>	78
---------------------------	----