People Scope API Document

Contents

Package PeopleScope Procedural Elements

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>
- **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>
- **Version** 1.0
- **Filesource** Source Code for this file

require_once **'errorhandler.class.php'** *[line 35]*

required error handler

department.class.php

- **Package** PeopleScope
- **Filesource** Source Code for this file

department.old.class.php

- **Package** PeopleScope
- **Filesource** Source Code for this file

Package PeopleScope Classes

Class department

[line 15]

Department Class

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

- **Package** PeopleScope
- **Author** Jennifer Erator < jason@lexxcom.com>
- **Version** 1.1 of the Framework generator

department::$db

Object = [line 27]

Database class object

- **Access** private

department::$db_connect

Object = [line 21]

Connect to PDO object through database class

- **Access** private

department::$fields

Array = array('dept_id', 'name', 'office_id') *[line 45]*

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

- **Access** private

department::$fields_required

Array|null = NULL *[line 51]*

Array of feilds require information when validating

- **Access** private

department::$fields_validation_type

Array|null = array ('dept_id'=>'INT', 'name'=>'TEXT', 'office_id'=>'INT') *[line 57]*

Array of feilds and there types that are check when validating

- **Access** private

department::$table

Object = [line 33]

Table class object

- **Access** public

department::$template

Object = [line 39]

Template class object

- **Access** public

department::$validation_error

Array = array() *[line 64]*

Array use to store any error found during Validation function

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

Constructor *void* function department::__construct() *[line 82]*

Contructor 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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department::$template
- **See** department::$table
- **See** db::
- **Access** public

Integer function department::create($source) *[line 174]*

Function Parameters:

- *Array* **$source**

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

This method will insert the formated 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
    )
)
```

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department::createDepartmentDetails() *[line 523]*

This method will provide a page to the to add a single row Department to the department 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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department::deleteDepartmentDetails($id) *[line 608]*

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department::editDepartmentDetails($id) *[line 437]*

Function Parameters:

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

Show the details ready to edit of a single Department from the department

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department::getDepartmentList([$type = 'TABLE'], [$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 363]*

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 Department
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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

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

Function Parameters:

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

Show will pull a list from the corresponding Department department

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

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department::read($id) *[line 220]*

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

NOTE: This is generated information from the framework and will need

to be corrected if there are any changes

- **Access** private

Boolean function department::remove($id) *[line 320]*

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department::saveDepartmentDetails($id) *[line 556]*

Function Parameters:

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

save the information in a single Department to the department table

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department::update()
- **See** department::Validate()
- **See** department::createDepartmentDetails()
- **Access** public

void function department::showDepartmentDetails($id) *[line 409]*

Function Parameters:

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

Show details of a single Department from the department

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department::templateDepartmentLayout($staffMember, [$input = false], [$inputArray = array()], $fielddata) *[line 627]*

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* **$inputArray** Not sure :S
- **$staffMember**

This method assigns the associate array values to the template

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

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

Integer function department::update($source, $id) *[line 273]*

Function Parameters:

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

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

This method will update the formated 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
    )
)
```

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department::updateDepartmentDetails($id) *[line 471]*

Function Parameters:

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

update the information in a single Department from the department

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 datebase 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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department::update()
- **See** department::Validate()
- **See** department::editDepartmentDetails()
- **Access** public

void function department::Validate($request) *[line 659]*

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department::$validation_error
- **See** department::$fields
- **See** department::$fields_validation_type
- **See** department::$fields_required
- **Access** public

Class department2

[line 17]

{$ucfClass}2 Class,

This class is based on the table {$table}

- **Package** PeopleScope
- **Author** Jason Stewart < jason@lexxcom.com.au>
- **Version** {$version}

  NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Version** {$version} of the Framework generator

department2::$db

Object = *[line 28]*

Database class object

- **Access** private

department2::$db_connect

Object = *[line 22]*

Connect to PDO object through database class

- **Access** private

department2::$fields

Array = array('dept_id', 'name', 'office_id') *[line 46]*

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

- **Access** private

department2::$fields_required

Array|null = NULL *[line 52]*

Array of feilds require information when validating

- **Access** private

department2::$fields_validation_type

Array|null = array ('dept_id'=>'INT', 'name'=>'TEXT', 'office_id'=>'INT') *[line 58]*

Array of feilds and there types that are check when validating

- **Access** private

department2::$table

Object = *[line 34]*

Table class object

- **Access** public

department2::$template

Object = *[line 40]*

Template class object

- **Access** public

department2::$validation_error

Array = array() *[line 64]*

Array use to store any error found during Validation function

- **Access** private

Constructor *void* function department2::__construct() *[line 82]*

Contructor 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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department2::$template
- **See** department2::$table
- **See** db::

Integer function department2::create($source) *[line 175]*

Function Parameters:

- *Array* **$source**

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

This method will insert the formated 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
)
)

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department2::createDepartmentDetails() *[line 517]*

This method will provide a page to the to add a single row {$ucfClass} to the {$table} 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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department2::deleteDepartmentDetails($id) *[line 601]*

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department2::editDepartmentDetails($id) *[line 431]*

Function Parameters:

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

Show the details ready to edit of a single {$ucfClass} from the {$table}

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
NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department2::getDepartmentList([$type = 'TABLE'], [$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 359]*

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=$val 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 {$ucfClass}
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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department2::lists([$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 116]*

Function Parameters:

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

Show will pull a list from the corresponding {$ucfClass} {$table}

This Method will produce a list of all the element corresponding to the result of {$ucfClass}

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department2::read($id) *[line 218]*

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

Boolean function department2::remove($id) *[line 317]*

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department2::saveDepartmentDetails($id) *[line 550]*

Function Parameters:

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

save the information in a single {$ucfClass} to the {$table} table

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department2::update()
- **See** department2::Validate()
- **See** department2::createDepartmentDetails()
- **Access** public

void function department2::showDepartmentDetails($id) *[line 404]*

Function Parameters:

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

Show details of a single {$ucfClass} from the {$table}

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** public

void function department2::templateDepartmentLayout($fielddata, [$input = false], [$inputArray = array()]) *[line 620]*

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

This method assigns the associate array values to the template

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

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

Integer function department2::update($source, $id) *[line 272]*

Function Parameters:

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

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

This method will update the formated 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
    )
)
```

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **Access** private

void function department2::updateDepartmentDetails($id) *[line 465]*

Function Parameters:

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

update the information in a single {$ucfClass} from the {$table}

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 datebase 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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department2::update()
- **See** department2::Validate()
- **See** department2::editDepartmentDetails()

- **Access** public

void function department2::Validate($request) *[line 651]*

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

NOTE: This is generated information from the framework and will need to be corrected if there are any changes

- **See** department2::$validation_error
- **See** department2::$fields
- **See** department2::$fields_validation_type
- **See** department2::$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>
- **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>
- **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>
- **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>
- **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 referance

Will convert the output from a Jquery UI date field format

format must be in Au date format

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

an function createDateField($date, [$yearRange = 10], [$startyear = NULL], $name) *[line 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 formatDateResponce($name, [&$listArray = NULL]) *[line 259]*
Function Parameters:

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

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

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

$name.'__day'
$name.'__month'
$name.'__year'

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

void function formatDateUI($date) *[line 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 debuging tool

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

void function showTrace() *[line 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* **$elements** Comma seperated list of names
- *Array* **$array** The array to remove them from via referance
- **&$array**

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

void function trace($var, [$newline = true]) *[line 68]*
Function Parameters:

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

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

- **See** showTrace for output

void function warning($content) *[line 438]*
Function Parameters:

- **$content**

$GLOBALS['style']

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

Global style for error messages

$GLOBALS['trace']

string = array() *[line 24]*

Global trace debugging

Class CustomException

[line 34]

Generates a custom exception

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

string function CustomException::logError() *[line 46]*

Generate a formated excption 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 227]*

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 instegate 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($table, $content, [$form_name = NULL]) *[line 196]*

Function Parameters:

- *String* **$table** table to store the email
- *String* **$content** content of the email
- *String* **$form_name** identifer of record, Default URI sent from

Record email into the database

void function email::send($to) *[line 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 varable converstion

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>
- **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>
- **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::genterateDisplayTable($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

Contructor for template class

- **Access** public

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

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

Contructor php4 for template class

- **Access** public

true function template::assign($name, $content) *[line 122]*
Function Parameters:

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

Used to assign a value element to a generated template

- **Access** public

true function template::assignArray($assignedArray) *[line 176]*
Function Parameters:

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

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

- **Access** public

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

Function Parameters:

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

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

example:

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

- **Access** public

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

Function Parameters:

- *String* **$name** Repeat assignment name
- *Array* **$content** Array of arrays values

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

```
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

Sting function template::BuildAssigned($content) *[line 202]*

Function Parameters:

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

Build template with assigned values

- **Access** private

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

Will print out the full generated page template

- **Access** public

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

Will return the full generated page template as a variarable

- **Access** public

true function template::replace($string, [$value = '']) *[line 190]*
Function Parameters:

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

Will replace the string across the entire template
replace happens before compiling, so it is possible to change a tag on the fly

- **Access** public

true function template::set($file) *[line 99]*
Function Parameters:

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

Set the current template path to a new template file

- **Access** public

Package default Procedural Elements

administration.class.php

- **Package** default
- **Filesource** Source Code for this file

advertisment.class.php

- **Package** default
- **Filesource** Source Code for this file

application.class.php

- **Package** default
- **Filesource** Source Code for this file

template.class.php

- **Package** default
- **Filesource** Source Code for this file

division.class.php

- **Package** default
- **Filesource** Source Code for this file

login.class.php

- **Package** default
- **Filesource** Source Code for this file

question.class.php

- **Package** default
- **Filesource** Source Code for this file

users.class.php

- **Package** default
- **Filesource** Source Code for this file

require_once **'division.class.php'** *[line 4]*

require_once **'administration.class.php'** *[line 3]*

Package default Classes

Class administration

[line 3]

- **Package** default

administration::$admin

mixed = [line 11]

- **Access** public

administration::$db

mixed = [line 6]

- **Access** private

administration::$db_connect

mixed = [line 5]

- **Access** private

administration::$error

mixed = *[line 10]*

- **Access** private

administration::$fields

mixed = array('administration_id', 'group_name', 'create_advert', 'edit_advert', 'remove_advert', 'create_template', 'edit_template', 'remove_template', 'add_user', 'edit_user', 'delete_user', 'edit_status', 'edit_referral') *[line 12]*

- **Access** private

administration::$fields_required

mixed = NULL *[line 13]*

- **Access** private

administration::$fields_validation_type

mixed = array ('administration_id'=>'INT', 'group_name'=>'TEXT', 'create_advert'=>'BOOL', 'edit_advert'=>'BOOL', 'remove_advert'=>'BOOL', 'create_template'=>'BOOL', 'edit_template'=>'BOOL', 'remove_template'=>'BOOL', 'add_user'=>'BOOL', 'edit_user'=>'BOOL', 'delete_user'=>'BOOL', 'edit_status'=>'BOOL', 'edit_referral'=>'BOOL') *[line 14]*

- **Access** private

administration::$lastError

mixed = [line 7]

- **Access** public

administration::$table

mixed = [line 8]

- **Access** public

administration::$template

mixed = [line 9]

- **Access** public

Constructor *void* function administration::__construct() *[line 16]*
void function administration::create($source) *[line 68]*
Function Parameters:

- **$source**

- **Access** private

void function administration::createAdministrationDetails() *[line 324]*

- **Access** public

void function administration::deleteAdministrationDetails($id) *[line 379]*

Function Parameters:

- **$id**

- **Access** public

void function administration::editAdministrationDetails($id) *[line 265]*

Function Parameters:

- **$id**

- **Access** public

void function administration::getAdministrationList([$type = 'TABLE'], [$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 196]*

Function Parameters:

- **$type**
- **$orderby**
- **$direction**
- **$filter**

********** END CRUD METHOD**********

- **Access** public

void function administration::getSelectListOfAdministration($id, [$selectBox = NULL]) *[line 497]*

Function Parameters:

- **$id**
- **$selectBox**

- **Access** public

void function administration::lists([$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 30]*

Function Parameters:

- **$orderby**
- **$direction**
- **$filter**

- **Access** private

void function administration::read($id) *[line 102]*

Function Parameters:

- **$id**

- **Access** private

void function administration::remove($id) *[line 178]*

Function Parameters:

- **$id**

  - **Access** private

void function administration::saveAdministrationDetails() *[line 339]*

  - **Access** public

void function administration::showAdministrationDetails($id, [$return = false]) *[line 251]*

Function Parameters:

- **$id**
- **$return**

  - **Access** public

void function administration::templateAdministrationLayout($staffMember, [$input = false], [$inputArray = array()]) *[line 384]*

Function Parameters:

- **$staffMember**
- **$input**
- **$inputArray**

  - **Access** private

void function administration::update($source, $id) *[line 134]*

Function Parameters:

- **$source**
- **$id**

- **Access** private

void function administration::updateAdministrationDetails($id) *[line 284]*

Function Parameters:

- **$id**

- **Access** public

void function administration::Validate($request) *[line 414]*

Function Parameters:

- **$request**

- **Access** public

Class advertisment

[line 3]

- **Package** default

advertisment::$admin

mixed = [line 11]

- **Access** public

advertisment::$db

mixed = [line 6]

- **Access** private

advertisment::$db_connect

mixed = [line 5]

- **Access** private

advertisment::$error

mixed = [line 10]

- **Access** private

advertisment::$fields

mixed = array('advertisment_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', 'create_date', 'create_by', 'modify_date', 'modify_by', 'delete_date', 'delete_by', 'question_id', 'tracking_id') *[line 12]*

- **Access** private

advertisment::$fields_required

mixed = array('title') *[line 13]*

- **Access** private

advertisment::$fields_validation_type

mixed = array ('advertisment_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'=>'INT', 'create_date'=>'TEXT', 'create_by'=>'INT', 'modify_date'=>'TEXT', 'modify_by'=>'INT', 'delete_date'=>'TEXT', 'delete_by'=>'INT', 'question_id'=>'INT', 'tracking_id'=>'INT') *[line 14]*

- **Access** private

advertisment::$lastError

mixed = *[line 7]*

- **Access** public

advertisment::$table

mixed = *[line 8]*

- **Access** public

advertisment::$template

mixed = [line 9]

- **Access** public

Constructor *void* function advertisment::__construct() *[line 16]*
void function advertisment::create($source) *[line 80]*
Function Parameters:

- **$source**

- **Access** private

void function advertisment::createAdvertismentDetails() *[line 382]*

- **Access** public

void function advertisment::deleteAdvertismentDetails($id) *[line 458]*
Function Parameters:

- **$id**

- **Access** public

void function advertisment::editAdvertismentDetails($id) *[line 304]*

Function Parameters:

- **$id**

- **Access** public

void function advertisment::getAdvertismentList([$type = 'TABLE'], [$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 209]*

Function Parameters:

- **$type**
- **$orderby**
- **$direction**
- **$filter**

********** END CRUD METHOD**********

- **Access** public

void function advertisment::lists([$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 30]*

Function Parameters:

- **$orderby**
- **$direction**
- **$filter**

- **Access** private

void function advertisment::read($id) *[line 114]*
Function Parameters:

- **$id**

- **Access** private

void function advertisment::remove($id) *[line 191]*
Function Parameters:

- **$id**

- **Access** private

void function advertisment::saveAdvertismentDetails() *[line 397]*

- **Access** public

void function advertisment::showAdvertismentDetails($id, [$return = false]) *[line 290]*
Function Parameters:

- **$id**
- **$return**

- **Access** public

void function advertisment::templateAdvertismentLayout($staffMember, [$input = false], [$inputArray = array()]) *[line 463]*

Function Parameters:

- **$staffMember**
- **$input**
- **$inputArray**

- **Access** private

void function advertisment::update($source, $id) *[line 158]*

Function Parameters:

- **$source**
- **$id**

- **Access** private

void function advertisment::updateAdvertismentDetails($id) *[line 321]*

Function Parameters:

- **$id**

- **Access** public

void function advertisment::Validate($request) *[line 501]*

Function Parameters:

- **$request**

- **Access** public

Class application

[line 3]

- **Package** default

application::$admin

mixed = [line 11]

- **Access** public

application::$db

mixed = [line 6]

- **Access** private

application::$db_connect

mixed = *[line 5]*

- **Access** private

application::$error

mixed = *[line 10]*

- **Access** private

application::$fields

mixed = array('application_id', 'applicant_id', 'job_id', 'catagory_id', 'viewed', 'contact_type_id', 'last_contact', 'referral_id', 'status_id', 'contact_susccessful', 'offer_successful', 'reciept_successful', 'saved', 'coverletter_file', 'coverletter_type', 'coverletter_text', 'resume_file', 'resume_type', 'create_date', 'modify_date', 'delete_date') *[line 12]*

- **Access** private

application::$fields_required

mixed = array('applicant_id', 'job_id', 'catagory_id') *[line 13]*

- **Access** private

application::$fields_validation_type

mixed = array ('application_id'=>'INT', 'applicant_id'=>'INT', 'job_id'=>'INT', 'catagory_id'=>'INT', 'viewed'=>'BOOL', 'contact_type_id'=>'INT', 'last_contact'=>'TEXT', 'referral_id'=>'INT', 'status_id'=>'INT', 'contact_susccessful'=>'BOOL', 'offer_successful'=>'BOOL', 'reciept_successful'=>'BOOL', 'saved'=>'BOOL', 'coverletter_file'=>'TEXT', 'coverletter_type'=>'TEXT', 'coverletter_text'=>'TEXT', 'resume_file'=>'TEXT', 'resume_type'=>'TEXT', 'create_date'=>'TEXT', 'modify_date'=>'TEXT', 'delete_date'=>'TEXT') *[line 14]*

- **Access** private

application::$lastError

mixed = [line 7]

- **Access** public

application::$table

mixed = [line 8]

- **Access** public

application::$template

mixed = [line 9]

- **Access** public

Constructor *void* function application::__construct() *[line 16]*
void function application::create($source) *[line 75]*
Function Parameters:

- **$source**

- **Access** private

void function application::createApplicationDetails() *[line 342]*

- **Access** public

void function application::deleteClientsDetails($id) *[line 396]*
Function Parameters:

- **$id**

- **Access** public

void function application::editApplicationDetails($id) *[line 284]*
Function Parameters:

- **$id**

- **Access** public

void function application::getApplicationList([$type = 'TABLE'], [$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 199]*
Function Parameters:

- **$type**
- **$orderby**
- **$direction**
- **$filter**

********** END CRUD METHOD**********

- **Access** public

void function application::lists([$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 30]*
Function Parameters:

- **$orderby**
- **$direction**
- **$filter**

- **Access** private

void function application::read($id) *[line 109]*
Function Parameters:

- **$id**

- **Access** private

void function application::remove($id) *[line 181]*
Function Parameters:

- **$id**

- **Access** private

void function application::saveApplicationDetails() *[line 357]*

- **Access** public

void function application::showApplicationDetails($id, [$return = false]) *[line 270]*

Function Parameters:

- **$id**
- **$return**

- **Access** public

void function application::templateApplicationLayout($staffMember, [$input = false], [$inputArray = array()]) *[line 401]*

Function Parameters:

- **$staffMember**
- **$input**
- **$inputArray**

- **Access** private

void function application::update($source, $id) *[line 148]*

Function Parameters:

- **$source**
- **$id**

- **Access** private

void function application::updateApplicationDetails($id) *[line 301]*
Function Parameters:

- **$id**

- **Access** public

void function application::Validate($request) *[line 434]*
Function Parameters:

- **$request**

- **Access** public

Class division
[line 3]

- **Package** default

division::$admin

mixed = *[line 11]*

- **Access** public

division::$db

mixed = *[line 6]*

- **Access** private

division::$db_connect

mixed = *[line 5]*

- **Access** private

division::$error

mixed = *[line 10]*

- **Access** private

division::$fields

mixed = array('division_id', 'name', 'description', 'create_date', 'modify_date', 'delete_date') *[line 12]*

- **Access** private

division::$fields_required

mixed = array('name') *[line 13]*

- **Access** private

division::$fields_validation_type

mixed = array ('division_id'=>'INT', 'name'=>'TEXT', 'description'=>'TEXT', 'create_date'=>'TEXT', 'modify_date'=>'TEXT', 'delete_date'=>'TEXT') *[line 14]*

- **Access** private

division::$lastError

mixed = *[line 7]*

- **Access** public

division::$table

mixed = *[line 8]*

- **Access** public

division::$template

mixed = *[line 9]*

- **Access** public

Constructor *void* function division::__construct() *[line 16]*
void function division::create($source) *[line 60]*
Function Parameters:

- **$source**

- **Access** private

void function division::createDivisionDetails() *[line 283]*

- **Access** public

void function division::deleteClientsDetails($id) *[line 338]*
Function Parameters:

- **$id**

- **Access** public

void function division::editDivisionDetails($id) *[line 225]*
Function Parameters:

- **$id**

- **Access** public

void function division::getDivisionList([$type = 'TABLE'], [$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 170]*

Function Parameters:

- **$type**
- **$orderby**
- **$direction**
- **$filter**

********** END CRUD METHOD**********

- **Access** public

void function division::getSelectListOfDivision($id, [$selectBox = NULL]) *[line 444]*

Function Parameters:

- **$id**
- **$selectBox**

- **Access** public

void function division::lists([$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 30]*

Function Parameters:

- **$orderby**
- **$direction**
- **$filter**

- **Access** private

void function division::read($id) *[line 95]*
Function Parameters:

- **$id**

- **Access** private

void function division::remove($id) *[line 152]*
Function Parameters:

- **$id**

- **Access** private

void function division::saveDivisionDetails() *[line 298]*

- **Access** public

void function division::showDivisionDetails($id, [$return = false]) *[line 211]*
Function Parameters:

- **$id**
- **$return**

- **Access** public

void function division::templateDivisionLayout($staffMember, [$input = false], [$inputArray = array()]) *[line 343]*
Function Parameters:

- **$staffMember**
- **$input**
- **$inputArray**

- **Access** private

void function division::update($source, $id) *[line 119]*
Function Parameters:

- **$source**
- **$id**

- **Access** private

void function division::updateDivisionDetails($id) *[line 242]*
Function Parameters:

- **$id**

- **Access** public

void function division::Validate($request) *[line 360]*

Function Parameters:

- **$request**

- **Access** public

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 57]*

Function Parameters:

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

- **Access** public

void function template::content($field) *[line 52]*
Function Parameters:

- **$field**

- **Access** public

void function template::display() *[line 75]*

- **Access** public

void function template::externalLink($link) *[line 155]*
Function Parameters:

- **$link**

- **Access** public

void function template::fetch([&$tpl = NULL]) *[line 66]*
Function Parameters:

- **&$tpl**

- **Access** public

void function template::formatBoolean($value) *[line 79]*

Function Parameters:

- **$value**

- **Access** public

void function template::formatValue($value, $msg) *[line 89]*

Function Parameters:

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

- **Access** public

void function template::getListTable($table, $value, $idField, $valueField, [$selectBox = NULL], [$WHERE = NULL]) *[line 99]*

Function Parameters:

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

- **Access** public

void function template::input($type, $name, [$value = NULL]) *[line 135]*

Function Parameters:

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

- **Access** public

void function template::insert($template) *[line 47]*
Function Parameters:

- **$template**

- **Access** public

void function template::page($field) *[line 41]*
Function Parameters:

- **$field**

- **Access** public

void function template::strip_tags($string) *[line 149]*
Function Parameters:

- **$string**

- **Access** private

void function template::__destruct() *[line 37]*

- **Access** public

Class users

[line 6]

- **Package** default

users::$admin

mixed = [line 16]

- **Access** public

users::$administration

mixed = [line 13]

- **Access** public

users::$db

mixed = [line 9]

- **Access** private

users::$db_connect

mixed = [line 8]

- **Access** private

users::$division

mixed = [line 14]

- **Access** public

users::$error

mixed = [line 15]

- **Access** private

users::$fields

mixed = array('user_id', 'username', 'password', 'name', 'surname', 'email', 'active', 'last_login', 'division_id', 'administration_id', 'create_date', 'modified_date', 'delete_date') *[line 17]*

- **Access** private

users::$fields_required

mixed = array('username', 'password', 'name', 'email') *[line 18]*

- **Access** private

users::$fields_validation_type

mixed = array ('user_id'=>'INT', 'username'=>'TEXT', 'password'=>'TEXT', 'name'=>'TEXT', 'surname'=>'TEXT', 'email'=>'TEXT', 'active'=>'BOOL', 'last_login'=>'TEXT', 'division_id'=>'INT', 'administration_id'=>'INT', 'create_date'=>'TEXT', 'modified_date'=>'TEXT', 'delete_date'=>'TEXT') *[line 19]*

- **Access** private

users::$lastError

mixed = *[line 10]*

- **Access** public

users::$table

mixed = *[line 11]*

- **Access** public

users::$template

mixed = *[line 12]*

- **Access** public

Constructor *void* function users::__construct() *[line 21]*
void function users::create($source) *[line 78]*
Function Parameters:

- **$source**

    - **Access** private

void function users::createUsersDetails() *[line 379]*

    - **Access** public

void function users::deleteClientsDetails($id) *[line 444]*
Function Parameters:

- **$id**

    - **Access** public

void function users::editUsersDetails($id) *[line 313]*
Function Parameters:

- **$id**

- **Access** public

void function users::getUsersList([$type = 'TABLE'], [$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 249]*

Function Parameters:

- **$type**
- **$orderby**
- **$direction**
- **$filter**

********** END CRUD METHOD**********

- **Access** public

void function users::lists([$orderby = NULL], [$direction = 'ASC'], [$filter = NULL]) *[line 37]*

Function Parameters:

- **$orderby**
- **$direction**
- **$filter**

- **Access** private

void function users::read($id) *[line 160]*

Function Parameters:

- **$id**

- **Access** private

void function users::remove($id) *[line 231]*
Function Parameters:

- **$id**

- **Access** private

void function users::saveUsersDetails() *[line 394]*

- **Access** public

void function users::showUsersDetails($id, [$return = false]) *[line 299]*
Function Parameters:

- **$id**
- **$return**

- **Access** public

void function users::templateUsersLayout($staffMember, [$input = false], [$inputArray = array()]) *[line 449]*
Function Parameters:

- **$staffMember**
- **$input**
- **$inputArray**

- **Access** private

void function users::update($source, $id) *[line 196]*

Function Parameters:

- **$source**
- **$id**

- **Access** private

void function users::updateUsersDetails($id) *[line 330]*

Function Parameters:

- **$id**

- **Access** public

void function users::Validate($request) *[line 470]*

Function Parameters:

- **$request**

- **Access** public

Appendices

Appendix A - Class Trees

Package default

administration

- administration

advertisment

- advertisment

application

- application

division

- division

login

- login

template

- template

users

- users

Package PeopleScope

CustomException

- Exception
  - CustomException

db

- db

department

- department

department2

- department2

email

- email

table

- table
  - form

template

- template

Appendix C - Source Code

Package default

File Source for errorhandler.class.php

Documentation for this file is available at 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:
"     .$this->   getLine();
38            return $str;
39        }
40
41        /**
42         * Generate a formated excption error
43         *
44         * @return string Html Format
45         */
46        public function logError(){
47
48            $str = "<div style=\"font-family: Arial, Helvetica, sans-serif; font-size:
15px; padding:10px; color:#000; background-color: #ffcccc; width:100%; border:solid 1px
#000\"><h4>CustomException Error Information</h4>"                                .
49                "<div> Message :: "                .$this->   getMessage()."</div>
50                <div>File ::"           .$this->   getFile()."</div>
51                <div>Line ::"           .$this->   getLine()."</div></pre>
52                </div>"           ;
53
54            if(DEBUG){
55                $trace = "<pre>"                .print_r($this-
>   getTrace(),1)."</pre>"                ;
56                $str = "<div style=\"font-family: Arial, Helvetica, sans-serif; font-size:
8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px #000\"><h4>Error
Back Trace</h4><pre>"                .
57                        htmlentities (print_r($trace,1),  ENT_COMPAT) .
58                        "</pre></div>"                    ;
59            }
60
61            return $str;
```

```
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"          .$line."\n"          ;
78              }
79              echo "<div style=\"font-family: Arial, Helvetica, sans-serif; font-size: 15px;
color:#000; padding:10px; background-color: #eecc00; width:100%; border:solid 1px
#000\"><h2>SQL Statement</h2><pre>"                                  .$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                     $trace = "<pre>"               .print_r($this-
>   getTrace(),1)."</pre>"               ;
115                     $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
#000\"><h4>Error Back Trace</h4><pre>"                                .
116                             htmlentities (print_r($trace,1),  ENT_COMPAT) .
117                             "</pre></div>"                    ;
118                 }
119                 return ($str);
120                 break;
121             case E_WARNING:
122             case E_CORE_WARNING:
123             case E_USER_WARNING:
124                 $str = "<div style=\"border: green 3px solid; font-family: Arial,
Helvetica, sans-serif; font-size: 15px; color:#000; padding:10px; background-color: #8AE234; width:100%;
border:solid 3px green\"><h4>PHP WARNING:: Error
Information["     .$errno."]</h4>"               .
125                 "<div> Message ::
<strong>"          .$errstr."</strong></div><br />
126                 <div>File :: "           .$errfile."</div>
127                 <div>Line :: "           .$errline."</div></pre>
128                 </div>"          ;
```

```
129             if(DEBUG){
130                 $trace = "<pre>"                .print_r($this-
>   getTrace(),1)."</pre>"              ;
131                 $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
#000\"><h4>Error Back Trace</h4><pre>"                                  .
132                         htmlentities (print_r($trace,1),  ENT_COMPAT) .
133                         "</pre></div>"                      ;
134             }
135             break;
136         case E_NOTICE:
137         case E_CORE_NOTICE:
138         case E_USER_NOTICE:
139             $str = "<div style=\"border: green 1px solid; font-family: Arial,
Helvetica, sans-serif; font-size: 15px; color:#000; padding:10px; background-color: #8AE234; width:100%;
border:solid 3px green\"><h4>PHP NOTICE:: Error
Information["     .$errno."]</h4>"               .
140             "<div> Message ::
<strong>"           .$errstr."</strong></div><br />
141             <div>File :: "           .$errfile."</div>
142             <div>Line :: "           .$errline."</div></pre>
143             </div>"           ;
144             if(DEBUG){
145                 $trace = "<pre>"                .print_r($this-
>   getTrace(),1)."</pre>"              ;
146                 $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
#000\"><h4>Error Back Trace</h4><pre>"                                  .
147                         htmlentities (print_r($trace,1),  ENT_COMPAT) .
148                         "</pre></div>"                      ;
149             }
150             break;
151
152         default:
153             $str = "<div style=\"border: green 1px solid; font-family: Arial,
Helvetica, sans-serif; font-size: 15px; color:#000; padding:10px; background-color: #8AE234; width:100%;
border:solid 3px green\"><h4>PHP OTHER:: Error
Information["     .$errno."]</h4>"               .
154             "<div> Message ::
<strong>"           .$errstr."</strong></div><br />
155             <div>File :: "           .$errfile."</div>
156             <div>Line :: "           .$errline."</div></pre>
157             </div>"           ;
158             if(DEBUG){
159                 $trace = "<pre>"                .print_r($this-
>   getTrace(),1)."</pre>"              ;
160                 $str .= "<div style=\"font-family: Arial, Helvetica, sans-serif; font-
size: 8px; padding:10px; background-color: #3399ff; width:100%; border:solid 1px
#000\"><h4>Error Back Trace</h4><pre>"                                  .
161                         htmlentities (print_r($trace,1),  ENT_COMPAT) .
162                         "</pre></div>"                      ;
163             }
164             break;
165     }
166     echo $str;
167     /* Don't execute PHP internal error handler */
168     return true;
169  }
```

File Source for template.old.class.php

Documentation for this file is available at [template.old.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',
'var4'=>'<strong>frank</strong>'});<br />
13     * <br />
14     * $ArrayVars[0]['name'] = 'john';<br />
15     * $ArrayVars[0]['age'] = '14';<br />
16     * $ArrayVars[1]['name'] = 'mary';<br />
17     * $ArrayVars[1]['age'] = '42';<br />
18     * $ArrayVars[2]['name'] = 'frank';<br />
19     * $ArrayVars[2]['age'] = '98';<br />
20     * <br />
21     * $template->assignRepeat('agelist', $ArrayVars);<br />
22     * $template->replace('../', '../../');<br />
23     * <br />
24     * $template->display();<br />
25     * <br />
26     *
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 Reapeat region
64         * @todo should be removed as we are not using Dreamweaver template anymore
65         * @var Array
66         */
```

```
67          var $repeatRegion=array();
68
69          /**
70           * Contructor 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           * Contructor 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          * bassed 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 debuging 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 assigment 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'
, $compiledtemplate, $content);
239                 }
240         }
241
242
243         if( preg_match( '/\{\*(.*?)\*\}/', $content ) ){
244
245             preg_match_all( '/\{\*(.*?)\*\}/' , $content, $variable);
246
247             foreach($variable[0] AS $key=>   $val){
248                 if (isset($this->   assigned[trim($variable[1][$key])])){
249                     $content =str_replace($variable[0][$key], $this-
>   assigned[trim($variable[1][$key])], $content );
250                 }
251             }
252         }
253
254         if( preg_match( '/\{\?(.*?)\?\}/', $content ) ){
255
256             preg_match_all( '/\{\?(.*?)\?\}/' , $content, $code);
257
258             foreach($code[0] AS $key=>   $val){
259                 ob_start();
260                 eval($code[1][$key]);
261                 $result = ob_get_contents();
262                 ob_clean();
263                 $content =str_replace($code[0][$key], $result, $content );
264             }
265         }
266
267         return $content;
268     }
269
270     /**
271      * Will return the full generated page template as a variarable
272      *
273      * @return Sting html Generated from gathered information
274      */
275     public function fetch(){
276         if($this->   template_file){
277             if(!$template = file_get_contents($this->   template_file)){
278                 throw new CustomException('$this->template_file: Unable to read file contents'   );
279             }
280
281             foreach($this->   replacer AS $key=>   $value){
282                 $template = str_replace($value['string'], $value['value'], $template);
283             }
284
285             $template=$this->   BuildAssigned($template);
286
287
288             if(DEBUG){
289                 $body = parseArray($template, '<body'    , ">"            );
290                 $template = str_replace($body, showVars(), $template);
291             }
292         }else{
293             if(!DEBUG){
294                 $vars = showVars();
295                 $template = '<html><body>'            .$vars.$this-
>   assigned['blank'].'</body></html>'            ;
296             }else{
297                 $template = $this->   assigned['blank'];
298             }
299         }
300         return $template;
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 department.class.php

Documentation for this file is available at department.class.php

```php
<?php
/**
 * Department Class
 * <pre>
 * This class is based on the table department
 *
 * <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>
 *
 * @author Jennifer Erator <jason@lexxcom.com>
 * @version 1.1 of the Framework generator
 * @package PeopleScope
 */

class department {

    /**
     * Connect to PDO object through database class
     * @var Object
     */
    private $db_connect;

    /**
     * Database class object
     * @var Object
     */
    private $db;

    /**
     * Table class object
     * @var Object
     */
    public $table;

    /**
     * Template class object
     * @var Object
     */
    public $template;

    /**
     * Array of field used in the database if not in this list is dropped from insert or update
     * @var Array
     */
    private $fields =array('dept_id', 'name', 'office_id');

    /**
     * Array of feilds require information when validating
     * @var Array|null
     */
    private $fields_required = NULL;

    /**
     * Array of feilds and there types that are check when validating
     * @var Array|null
     */
    private $fields_validation_type = array ('dept_id'=>   'INT', 'name'=>   'TEXT',
'office_id'=>   'INT');

    /**
     * Array use to store any error found during Validation function
     * @see Validation()
     * @var Array
     */
    private $validation_error = array();

```

```
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        /**
97         * Show will pull a list from the corresponding Department department
98         *
99         * <pre>
100        * This Method will produce a list of all the element corresponding to the result of Department
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           $sql = "SELECT dept_id,
118   name,
119   office_id FROM department WHERE (delete_date ='00-00-0000 00:00:00' OR delete_date IS NULL)"     ;
120
121           if(is_array($filter)){
122                 foreach($filter AS $key=>   $value){
123                     if ($value != 'NULL'  && !empty(        $value)){
124                         $sql .=  " AND "          . $value;
125                     }
126                 }
127           }
128
129           if($orderby){
130                 $sql .= " ORDER BY "          . $orderby." "          .$direction;
131           }
132
133           try{
134                $result = $this->   db->   select($sql);
135           }catch(CustomException $e){
136                echo $e->   queryError($sql);
137           }
138
139           return $result;
140       }
141
142       /**
```

```
143         * This method will take an array and insert it in the database
144         *
145         * <pre>
146         * This method will insert the formated information into a database, the format for the array
147         * should be an associated array being the first key should be the table inserting with the keys
148         * for child array the fields that are being inserted too and the values to insert
149         *
150         * Array
151         *(
152         *    [users] => Array
153         *        (
154         *            [name] => Dave
155         *            [surname] => Smith
156         *            [email] => dave@dave.com
157         *        )
158         *    [staff] => Array
159         *        (
160         *            [staff_id] => 1245
161         *            [office_number] => 22
162         *            [drown_code] => bee223
163         *        )
164         * )
165         *
166         * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
167         *
168         *</pre>
169         *
170         * @param Array $source
171         *
172         * @return Integer Return last inserted primary id
173         */
174        Private function create($source){
175                try{
176                    $this->   db_connect->   beginTransaction();
177
178                    foreach($source['department'] AS $key=>   $val){
179                        $field[] = $key;
180                        $value[] = ":"          .$key;
181                    }
182
183                    $sql = "INSERT INTO department ("          .implode(', ',$field).") VALUES
("     .implode(', ',$value).");"          ;
184
185                    foreach($source['department'] AS $key=>   $val){
186                        $exec[":"          .$key] = $val;
187                    }
188
189                    try{
190                        $pid = $this->   db->   insert($sql, $exec);
191                    }catch(CustomException $e){
192                        throw new CustomException($e->   queryError($sql));
193                    }
194                    $this->   db_connect->   commit();
195                    $pid = $this->   db_connect->   lastInsertId();
196                }
197
198                catch (CustomException $e) {
199                    $e->   queryError($sql);
200                    $this->   db_connect->   rollBack();
201                    return false;
202                }
203
204                return $pid;
205        }
206
207
208        /**
209         * This method will return information as row
210         *
211         * <pre>
212         * This method is you to get a single row of information from the database
213         * based ith the primary id and return it as an array
214         *
215         * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
216         * </pre>
217         *
218         * @param Integer $id The primary id of the row to show
219         */
```

```
220       Private function read($id){
221
222           $sql = "SELECT dept_id,
223   name,
224   office_id FROM department WHERE dept_id = "     . $id ." AND (delete_date ='00-00-0000
00:00:00' OR delete_date IS NULL)"      ;
225
226
227               $stmt = $this->   db_connect->   prepare($sql);
228               $stmt->   execute();
229
230               try{
231                    $result = $this->   db->   select($sql);
232               }catch(CustomException $e){
233                    echo $e->   queryError($sql);
234               }
235
236               return $result[0];
237
238
239       }
240
241       /**
242        * This method will take an array and update a row in the database
243        *
244        * <pre>
245        * This method will update the formated information into the database, the format for the array
246        * should be an associated array being the first key should be the table to be updated with the
keys
247        * for child array the fields that are being updated too and the values to be updated
248        *
249        * Array
250        *(
251        *    [users] => Array
252        *        (
253        *            [name] => Dave
254        *            [surname] => Smith
255        *            [email] => dave@dave.com
256        *        )
257        *    [staff] => Array
258        *        (
259        *            [staff_id] => 1245
260        *            [office_number] => 22
261        *            [drown_code] => bee223
262        *        )
263        * )
264        *
265        * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
266        *
267        *</pre>
268        *
269        * @param Array $source
270        *
271        * @return Integer Return last inserted primary id
272        */
273       Private function update($source, $id){
274               try{
275                   $this->   db_connect->   beginTransaction();
276
277                   foreach($source['department'] AS $key=>   $val){
278                       $field[] = $key." = :"          .$key;
279                   }
280
281                   $sql = "UPDATE department SET "          .implode(', ',$field)." WHERE dept_id
="     . $id;
282
283                   foreach($source['department'] AS $key=>   $val){
284                       $exec[":"          .$key] = $val;
285                   }
286
287                   try{
288                           $pid = $this->   db->   update($sql, $exec);
289                   }catch(CustomException $e){
290                           throw new CustomException($e->   queryError($sql));
291                   }
292                   $this->   db_connect->   commit();
293                   $pid = $this->   db_connect->   lastInsertId();
294               }
295
```

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

```
372                             $this->   table->   setIdentifier('dept_id');
373                             $this->   table->   setIdentifierPage('department');
374                             echo $this->   table->   genterateDisplayTable($result);
375
376                 BREAK;
377             case 'TABLE' :
378             DEFAULT :
379                 $this->   table->   setHeader(array(
380                             'dept_id'=>   'Dept Id',
381   'name'=>   'Name',
382   'office_id'=>   'Office Id'));
383
384                 $this->   table->   setFilter(array(
385                             'dept_id'=>   'TEXT',
386   'name'=>   'TEXT',
387   'office_id'=>   'TEXT'));
388
389                 $this->   table->   setIdentifier('dept_id');
390
391                 $this->   template->   content(Box($this->   table-
>   genterateDisplayTable($result),'Department List', 'Shows the current listings for the Department'));
392
393                 $this->   template->   display();
394             }
395         }
396
397
398         /**
399          * Show details of a single Department from the department
400          *
401          * <pre>This method will return a template page of the information requested
402          * the method use the template class to format the information ready to display the
403          * the user
404          *
405          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
406          * </pre>
407          * @param Integer $id the primary id of the row to show
408          */
409         Public function showDepartmentDetails($id){
410             $staffMember = $this->   read($id);
411
412             $this->   template->   page('department.tpl.html');
413
414             $this->   templateDepartmentLayout($staffMember);
415
416             //if($this->checkAdminLevel(1)){
417                 $this->   template->   assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='department.php?action=edit&id="             .$id."'\">Edit</div&
gt;"     );
418             //}
419
420             echo $this->   template->   fetch();
421         }
422
423
424         /**
425          * Show the details ready to edit of a single Department from the department
426          *
427          * <pre>
428          * This method is used to display and editable page to the use, so that they
429          * maybe able to edit any of the fields releated to the row in question.
430          * The method uses the template class to format the information ready to display the
431          * the user
432          *
433          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
434          * </pre>
435          * @param Integer $id The primary id of the row to show
436          */
437         Public function editDepartmentDetails($id){
438
439             $staffMember = $this->   read($id);
440
441             $name = 'editDepartment';
442
443             $this->   template->   page('department.tpl.html');
444             $this->   template->   assign('FORM-HEADER', '<form
action="department.php?action=update&id='         .$id.'" method="POST"
name="'     .$name.'">'         );
```

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

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

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

```
any that are not
647         * 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
648         * 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
649         * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
650         * </pre>
651         *
652         * @see fields
653         * @see fields_required
654         * @see fields_validation_type
655         * @see validation_error
656         *
657         * @param Array $request
658         */
659        public function Validate($request){
660
661            unset($this->   valid_field);
662            unset($this->   validation_error);
663            $isvalid = True;
664
665            $validfields = $this->   fields;
666            $requiredfields = $this->   fields_required;
667            $fieldsvalidationtype = $this->   fields_validation_type;
668
669            foreach ($request AS $key=>   $value){ //lets strip put unwanted or security violation
fields
670                if(in_array($key, $validfields)){
671                    $this->   valid_field[$key] = $value; //pure fields
672                }
673            }
674
675            foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxs
676                if(!isset($this->   valid_field[$value])){
677                    $this->   valid_field[$value] = '';
678                }
679            }
680
681            if(count($requiredfields) >    0 ){
682                foreach($requiredfields AS $value){ // lets check all the required fields have a value
683                    if (empty($this->   valid_field[$value]) || $this->   valid_field[$value] ==
'NULL'){
684                        $this->   validation_error[$value][] = 'Field is Required'; //error field
685                        $isvalid = false;
686                    }
687                }
688            }
689
690
691
692            //now lets validate
693            foreach ($this->   valid_field AS $key=>   $value){
694                $value = trim($value);
695                if(!empty($value)){ // don't cheak if empty, alread done in required check
696
697                    switch(@$fieldsvalidationtype[$key]){
698                        case 'TEXTAREA': if (strlen($value) >    1024) {
699                                        $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
700                                        $isvalid = false;
701                                    } break;
702                        case 'TEXT': if (strlen($value) >    1024) {
703                                        $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
704                                        $isvalid = false;
705                                    } break;
706                        case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
707                                        $this->   validation_error[$key][] = 'not a valid SAP number';
708                                        $isvalid = false;
709                                    } break;
710                        case 'DECIMAL': if (!is_numeric($value)) {
711                                        $this->   validation_error[$key][] = 'Decimal value expected';
712                                        $isvalid = false;
713                                    } break;
714                        case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"          ) && (        $value != 1)) {
715                                        $this->   validation_error[$key][] = 'Please check';
716                                        $isvalid = false;
```

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

File Source for department.old.class.php

Documentation for this file is available at department.old.class.php

```
1     <?php
2     /**
3      * {$ucfClass}2 Class,
4      * <br />
5      * This class is based on the table {$table}
6      *
7      * @version {$version}
8      *
9      *  <div style="color:red">NOTE: This is generated information from the framework
and will need to be corrected if there are any changes</div>
10     *  <br />
11     *
12     * @author Jason Stewart <jason@lexxcom.com.au>
13     * @version {$version} of the Framework generator
14     * @package PeopleScope
15     */
16
17    class department2 {
18        /**
19         * Connect to PDO object through database class
20         * @var Object
21         */
22        private $db_connect;
23
24        /**
25         * Database class object
26         * @var Object
27         */
28        private $db;
29
30        /**
31         * Table class object
32         * @var Object
33         */
34        public $table;
35
36        /**
37         * Template class object
38         * @var Object
39         */
40        public $template;
41
42        /**
43         * Array of field used in the database if not in this list is dropped from insert or update
44         * @var Array
45         */
46        private $fields =array('dept_id', 'name', 'office_id');
47
48        /**
49         * Array of feilds require information when validating
50         * @var Array|null
51         */
52        private $fields_required = NULL;
53
54        /**
55         * Array of feilds and there types that are check when validating
56         * @var Array|null
57         */
58        private $fields_validation_type = array ('dept_id'=>   'INT', 'name'=>   'TEXT',
'office_id'=>   'INT');
59
60        /**
61         * Array use to store any error found during Validation function
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          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              $this->   template = new template();
93
94          }
95
96          /**
97           * Show will pull a list from the corresponding {$ucfClass} {$table}
98           *
99           * <pre>
100          * This Method will produce a list of all the element corresponding to the result of {$ucfClass}
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
116         Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
117
118             $sql = "SELECT dept_id,
119     name,
120     office_id FROM department WHERE (delete_date ='00-00-0000 00:00:00' OR delete_date IS NULL)"     ;
121
122             if(is_array($filter)){
123                   foreach($filter AS $key=>   $value){
124                       if ($value != 'NULL'  && !empty(        $value)){
125                           $sql .=  " AND "          . $value;
126                       }
127                   }
128             }
129
130             if($orderby){
131                   $sql .= " ORDER BY "          . $orderby." "          .$direction;
132             }
133
134             try{
135                 $result = $this->   db->   select($sql);
136             }catch(CustomException $e){
137                   echo $e->   queryError($sql);
138             }
139
140             return $result;
141         }
142
```

```
143         /**
144          * This method will take an array and insert it in the database
145          *
146          * <pre>
147          * This method will insert the formated information into a database, the format for the array
148          * should be an associated array being the first key should be the table inserting with the keys
149          * for child array the fields that are being inserted too and the values to insert
150          *
151          * Array
152          *(
153          *    [users] => Array
154          *        (
155          *            [name] => Dave
156          *            [surname] => Smith
157          *            [email] => dave@dave.com
158          *        )
159          *    [staff] => Array
160          *        (
161          *            [staff_id] => 1245
162          *            [office_number] => 22
163          *            [drown_code] => bee223
164          *        )
165          * )
166          *
167          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
168          *
169          *</pre>
170          *
171          * @param Array $source
172          *
173          * @return Integer Return last inserted primary id
174          */
175         Private function create($source){
176                 try{
177                     $this->   db_connect->   beginTransaction();
178
179                     foreach($source['department'] AS $key=>   $val){
180                         $field[] = $key;
181                         $value[] = ":"          .$key;
182                     }
183
184                     $sql = "INSERT INTO department ("          .implode(', ',$field).") VALUES
("      .implode(', ',$value).");"          ;
185
186                     foreach($source['department'] AS $key=>   $val){
187                         $exec[":"          .$key] = $val;
188                     }
189
190                     try{
191                         $pid = $this->   db->   insert($sql, $exec);
192                     }catch(CustomException $e){
193                         throw new CustomException($e->   queryError($sql));
194                     }
195                     $this->   db_connect->   commit();
196                     $pid = $this->   db_connect->   lastInsertId();
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          * This method will return information as row
210          * <pre>
211          * This method is you to get a single row of information from the database
212          * based ith the primary id and return it as an array
213          *
214          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
215          * </pre>
216          * @param Integer $id The primary id of the row to show
217          */
218         Private function read($id){
219
```

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

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

```
372                 BREAK;
373             case 'TABLE' :
374             DEFAULT :
375                 $this->   table->   setHeader(array(
376                         'dept_id'=>   'Dept Id',
377   'name'=>   'Name',
378   'office_id'=>   'Office Id'));
379
380                 $this->   table->   setFilter(array(
381                         'dept_id'=>   'TEXT',
382   'name'=>   'TEXT',
383   'office_id'=>   'TEXT'));
384
385                 $this->   table->   setIdentifier('dept_id');
386
387                 $this->   template->   content(Box($this->   table-
>   genterateDisplayTable($result),'Department List', 'Shows the current listings for the Department'));
388
389                 $this->   template->   display();
390         }
391     }
392
393     /**
394      * Show details of a single {$ucfClass} from the {$table}
395      *
396      * <pre>This method will return a template page of the information requested
397      * the method use the template class to format the information ready to display the
398      * the user
399      *
400      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
401      * </pre>
402      * @param Integer $id the primary id of the row to show
403      */
404     Public function showDepartmentDetails($id){
405         $staffMember = $this->   read($id);
406
407         $this->   template->   page('department.tpl.html');
408
409         $this->   templateDepartmentLayout($staffMember);
410
411         //if($this->checkAdminLevel(1)){
412             $this->   template->   assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='department.php?action=edit&id="             .$id."'\">Edit</div&
gt;"     );
413         //}
414
415         echo $this->   template->   fetch();
416     }
417
418
419     /**
420      * Show the details ready to edit of a single {$ucfClass} from the {$table}
421      *
422      * <pre>
423      * This method is used to display and editable page to the use, so that they
424      * maybe able to edit any of the fields releated to the row in question.
425      * The method uses the template class to format the information ready to display the
426      * the user
427      * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
428      * </pre>
429      * @param Integer $id The primary id of the row to show
430      */
431     Public function editDepartmentDetails($id){
432
433         $staffMember = $this->   read($id);
434
435         $name = 'editdepartment';
436
437         $this->   template->   page('department.tpl.html');
438         $this->   template->   assign('FORM-HEADER', '<form
action="department.php?action=update&id='         .$id.'" method="POST"
name="'     .$name.'">'        );
439
440         $this->   templateDepartmentLayout($staffMember, true);
441
442         $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Update</div><div
class=\"button\"
```

```
onclick=\"location.href='department.php?action=show&id=         "     .$id."'\">Cancel</di
v>"        );
443
444             $this->   template->   display();
445         }
446
447         /**
448          * update the information in a single {$ucfClass} from the {$table}
449          *
450          * <pre>
451          * This method is used to take the information from editDepartmentDetails and try to validate
it thought the
452          * validation method and on success will format it ready for input into the datebase through
the update method
453          *
454          * if the validate fails then the user is show a page that mimics the editDepartmentDetails
method and point out
455          * error in there input
456          *
457          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
458          * </pre>
459          *
460          * @see editDepartmentDetails()
461          * @see Validate()
462          * @see update()
463          * @param Integer $id The primary id of the row to updated
464          */
465         Public function updateDepartmentDetails($id){
466
467             if ($this->   Validate($_REQUEST)){
468
469                     $request = $_REQUEST;
470                     $table = 'department';
471
472                     $save[$table]['name'] = $request['name'];
473                     $save[$table]['office_id'] = $request['office_id'];
474
475                     //$save[$table]['modify_date'] = date('Y-m-d h:i:s');
476
477                     $this->   update($save, $id );
478
479                 }else{
480
481                     $staffMember = $this->   valid_field;
482                     $error = $this->   validation_error;
483
484                     $name = 'editgrant';
485
486                     $this->   template->   page('department.tpl.html');
487
488                     foreach($error AS $key=>   $value){
489                         $this->   template->   assign('err_'.$key, "<span
class=\"error\">"                    .@implode(',', $error[$key])."</spam>"                );
490                     }
491
492                     $this->   template->   assign('FORM-HEADER', '<form
action="department.php?action=update&id='         .$id.'" method="POST"
name="'     .$name.'">'        );
493
494                     $this->   templateDepartmentLayout($staffMember, true);
495
496                     //if($this->admin->checkAdminLevel(1)){
497                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.                $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='department.php?action=show&id=         "     .$id."'\">Cancel</di
v>"        );
498                     //}
499                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
500
501                     $this->   template->   display();
502             }
503         }
504
505         /**
506          * This method will provide a page to the to add a single row {$ucfClass} to the {$table} table
507          *
508          * <pre>
509          * The method using the template class to format the information ready to display the
```

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

```
action="department.php?action=save" method="POST" name="'                    .$name.'">'        );
579
580                     $this->   templateDepartmentLayout($staffMember, true);
581
582                     //if($this->admin->checkAdminLevel(1)){
583                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.                $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='department.php\">Cancel</div>                    "     );
584                     //}
585                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
586
587                     $this->   template->   display();
588             }
589         }
590
591         /**
592          * Set a row to be marked as deleted
593          *
594          * <pre>
595          * This method will take the id and set the delete_date field to
596          * the current datetime, which will marking it as deleted
597          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
598          * </pre>
599          * @param Integer $id The primary id of the row to marked as delete
600          */
601         Public function deleteDepartmentDetails($id){
602             $this->   remove($id);
603             header('Location: department.php');
604         }
605
606         /**
607          * This method assigns the associate array values to the template
608          *
609          * <pre>
610          * This method is used to incorperate the standards elements of the templates to a single
611          * function across all tempatled methods
612          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
613          * </pre>
614          * @todo find out what $inputArray is used for
615          *
616          * @param Array $fielddata An associative array of fields that need to be assigned to the
template object
617          * @param Boolean $input If false then just assign the value if true the add the value to
corresponding form element
618          * @param Array $inputArray Not sure :S
619          */
620         private function templateDepartmentLayout($fielddata, $input = false, $inputArray=array() ){
621
622                     $id = $fielddata['dept_id'];
623
624                     @$this->   template->   assign('dept_id', ($input)? $this->   template-
>   input('text', 'dept_id', $fielddata['dept_id']):$fielddata['dept_id']);
625                     @$this->   template->   assign('name', ($input)? $this->   template-
>   input('text', 'name', $fielddata['name']):$fielddata['name']);
626                     @$this->   template->   assign('office_id', ($input)? $this->   template-
>   input('text', 'office_id', $fielddata['office_id']):$fielddata['office_id']);
627
628
629                     /*if(isset($id)){
630                         $this->template->assign('COMMENTS', $this->comment-
>getCommentBox($id, 'department'));
631                     }*/
632
633         }
634
635         /**
636          * This medthod is used to validate inputs from form information
637          * <pre>
638          * This method will first check the if the fields are in the valid_field array and strip out
any that are not
639          * 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
640          * 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
641          * <div style="color:red">NOTE: This is generated information from the
framework and will need to be corrected if there are any changes</div>
642          * </pre>
```

```
643         *
644         * @see fields
645         * @see fields_required
646         * @see fields_validation_type
647         * @see validation_error
648         *
649         * @param Array $request
650         */
651        public function Validate($request){
652
653            unset($this->   valid_field);
654            unset($this->   validation_error);
655            $isvalid = True;
656
657            $validfields = $this->   fields;
658            $requiredfields = $this->   fields_required;
659            $fieldsvalidationtype = $this->   fields_validation_type;
660
661            foreach ($request AS $key=>   $value){ //lets strip put unwanted or security violation
fields
662                if(in_array($key, $validfields)){
663                    $this->   valid_field[$key] = $value; //pure fields
664                }
665            }
666
667            foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxs
668                if(!isset($this->   valid_field[$value])){
669                    $this->   valid_field[$value] = '';
670                }
671            }
672
673            if(count($requiredfields) >    0 ){
674                foreach($requiredfields AS $value){ // lets check all the required fields have a value
675                    if (empty($this->   valid_field[$value]) || $this->   valid_field[$value] ==
'NULL'){
676                        $this->   validation_error[$value][] = 'Field is Required'; //error field
677                        $isvalid = false;
678                    }
679                }
680            }
681
682
683
684            //now lets validate
685            foreach ($this->   valid_field AS $key=>   $value){
686                $value = trim($value);
687                if(!empty($value)){ // don't cheak if empty, alread done in required check
688
689                    switch(@$fieldsvalidationtype[$key]){
690                        case 'TEXTAREA': if (strlen($value) >    1024) {
691                                            $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
692                                            $isvalid = false;
693                                        } break;
694                        case 'TEXT': if (strlen($value) >    1024) {
695                                            $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
696                                            $isvalid = false;
697                                        } break;
698                        case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
699                                            $this->   validation_error[$key][] = 'not a valid SAP number';
700                                            $isvalid = false;
701                                        } break;
702                        case 'DECIMAL': if (!is_numeric($value)) {
703                                            $this->   validation_error[$key][] = 'Decimal value expected';
704                                            $isvalid = false;
705                                        } break;
706                        case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"          ) && (        $value != 1)) {
707                                            $this->   validation_error[$key][] = 'Please check';
708                                            $isvalid = false;
709                                        } break;
710                        case 'INT': if (!is_numeric($value) &&         $value != 'NULL' ){
711                                            $this->   validation_error[$key][] = 'Numeric value expected';
712                                            $isvalid = false;
713                                        } break;
714                        case 'DATE': $date = str_replace('/', '-', $value);
715                                     $date = str_replace("\\"          , '-', $date);
716                                        @list($day, $month, $year) = explode('-', $date);
```

```
717                                     if(!checkdate($month,$day, $year)){
718                                         $this->   validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
719                                         $isvalid = false;
720                                     } break;
721                         case 'YEAR':  if(!checkYear($value)){
722                                         $this->   validation_error[$key][] = 'incorrect year
format, expecting yyyy';
723                                         $isvalid = false;
724                                     } break;
725
726                     }
727                 }
728             }
729
730             return $isvalid;
731
732         }
733     }
```

File Source for database.class.php

Documentation for this file is available at database.class.php

```php
<?php
/**
 * Database Class,
 * <br />
 * This class is Database abstraction layer<br />
 *
 * Example:<br />
 * <br />
 * define('DB_USER','root');<br />
 * define('DB_PASS','password');<br />
 * define('DB_HOST','localhost');<br />
 * define('DB_DBASE','test_db');<br />
 * define('DB_TYPE','mysql');<br />
 * <br />
 * try{$db = new db();}<br />
 * catch(CustomException e){ echo $e->logError(); }<br />
 * <br />
 * try{$result = $db->select('select * from test_table');}<br />
 * catch(CustomException e){echo $e->queryError();}<br />
 * <br />
 * foreach($result AS $row){<br />
 *         echo 'col1 ='.$row['col1'];<br />
 *         echo 'col2 ='.$row['col2'];<br />
 *        echo 'col3 ='.$row['col3'];<br />
 * }<br />
 * <br />
 *
 * @author Jason Stewart <jason@lexxcom.com.au>
 * @version 1.0
 * @package PeopleScope
 * @subpackage Base
 */

if (isset($_SESSION['dbaccess']['server_host'])) {
    $DB_HOST = $_SESSION['dbaccess']['server_host'];
}else{
    $DB_HOST = DB_HOST;
}

if (isset($_SESSION['dbaccess']['server_database'])) {
    $DB_DBASE = $_SESSION['dbaccess']['server_database'];
}else{
    $DB_DBASE = DB_DBASE;
}

if (isset($_SESSION['dbaccess']['server_type'])) {
    $DB_TYPE = strtolower($_SESSION['dbaccess']['server_type']);
}else{
    $DB_TYPE = DB_TYPE;
}

if (isset($_SESSION['dbaccess']['server_port'])) {
    $DB_TYPE = $_SESSION['dbaccess']['server_port'];
}else{
    $DB_PORT = DB_PORT;
}

/**
 * required general tool
 */
require_once 'tools.function.php';
/**
 * required error handler
 */
require_once 'errorhandler.class.php';


```

```php
/**
 * This class is Database abstraction layer
 *
 * @package PeopleScope
 * @subpackage Base
 */
class db {

    var $dsn;
    var $dbh;
    var $lastQuery;


    /**
     * constructor to initiate database conection
     *
     * @return Void
     */
    function __construct(){

        $this->   dsn = DB_TYPE.':dbname='.DB_DBASE.';host='.DB_HOST.';port='.DB_PORT;
        try{
            try {
                $this->   dbh = new PDO($this->   dsn, DB_USER, DB_PASS);
            } catch (PDOException $e) {
                throw new CustomException('Connection Error '.$e->   getMessage());
            }
        }catch(CustomException $e) {
            echo $e->   logError();
        }
    }

    /**
     * returns current DSN string used to connect ot the server
     *
     * @return String
     */
    public function getDNSString(){
        return $this->   dsn;
    }

    /**
     * Will instigate a SELECT query and return and an array of responses
     *
     * @param String $sql select sql string to be executed
     * @return Array
     */
    public function select($sql){

        try{
            $stmt = $this->   query($sql);
        }catch(CustomException $e){
            throw new CustomException('SELECT : '. $e->   getMessage());
        }

        return $stmt->   fetchAll(PDO::FETCH_NAMED);

    }

    /**
     * Will instigate a INSERT query and return the inserts autocomplete Id;
     *
     * @param String $sql
     * @return Array
     */
    public function insert($sql, $exec = NULL){

        try{
            $stmt = $this->   query($sql, $exec);
        }catch(CustomException $e){
            throw new CustomException('INSERT : '. $e->   getMessage());
        }

        return $this->   dbh->   lastInsertId();

    }

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

```
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 instegate 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                    $error .= $stmt->   debugDumpParams();
207                }
208                  $error = $stmt->   errorInfo();
209                  throw new CustomException($error[2]);
210            }
211
212            return $stmt;
213        }
214
215        /**
216         * Merge query template with array
217         *
218         * Query template and array set merger function
219         * Will taken in the Query string template and the array of query elements
220         * and combine the to show the fully converted string used in the prepare
221         * Note: only use as example for debugging purposes
222         *
223         * @param string $query Query template
224         * @param array $params Array of inputs
225         * @return string
226         */
227        public function prepareToQuery($query, $params){
```

```
228             $keys = array();
229
230             # build a regular expression for each parameter
231             foreach ($params as $key =>    $value) {
232                 if (is_string($key)) {
233                     $keys[] = '/'.$key.'/';
234                 } else {
235                     $keys[] = '/[?]/';
236                 }
237             }
238
239             $query = preg_replace($keys, $params, $query, 1, $count);
240
241             return $query;
242         }
243
244     }
245
246     ?>
```

File Source for email.class.php

Documentation for this file is available at email.class.php

```
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 $content array of assigned varable converstion
112          * @see template::assignArray
113          */
114         function setHTMLtemplate($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             $str = '';
134             foreach ($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,$field) = explode(':', $info);
141
142                 $str .= $lable. "= "          ;
143                 if (isset($content[$field])){
144                     if (is_array($content[$field])){
145                         $str .=  implode(","           , $content[$field]) ."\n"          ;
146                     }else{
```

```
147                     $str .=  $content[$field] ."\n"          ;
148                 }
149             }
150         }
151
152         $this->   mime->   setTXTBody($str);
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 $form_name identifer of record, Default URI sent from
195      */
196     function recordEmail($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 ,
"subject_line"          =>   $this->   header['Subject'], "form_name"          =>   $form_name,
"create_date"          =>   $create_date);
208
209         $contentValueArray = array_merge($content, $set_values);
210         $fieldcount = count($contentValueArray);
211         $columnsArray = array_keys($contentValueArray);
212         $columnStr = implode(',', $columnsArray);
213
214         $sql = "INSERT INTO "          . $table ." ("          .$columnStr.") values
("     .str_repeat  ( "?, "          , $fieldcount-1 )."?);"          ;
215
216         $recordinsert = $this->   dbase->   Prepare($sql);
217         $ok = $this->   dbase->   Execute($recordinsert, $contentValueArray);
218
219         if (!$ok) {
220             pp($this->   dbase->   ErrorMsg(),1);
221             return false;
222         }
223
```

```
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

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

```
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          * Consrtructor
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          * Consrtructor 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 referance 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             $ret .=($this->   enctype)? ' enctype="'     .$this->   enctype.'"'      : '';
188             $ret .=($this->   formScript)? ' onSubmit="'     .$this->   formScript.'"'      : '';
189             $ret .= '>'   ;
190 
191             return $ret;
192         }
193 
194         /**
195          * Setup closing form tag
196          *
197          * @return String
198          */
199         function formFooter(){
200 
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 $column Define which column shold be returned
211          * @return String Html version on the generated input fields
212          */
213         function draw($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
226             $count=0;
227             //Null Error
228             $columnBreakDown = @explode("\n"          , $columnList[$column]);
229
230             foreach($columnBreakDown AS $value){
231                 if (!empty($value)){
232                     $inputfield[] = $this->   inputfield($value);
233                 }
234             }
235             return @implode("\n"          , $inputfield);
236         }
237
238         /**
239          * Set a alternate FCK config file path
240          *
241          * @param String $url Url to new config file
242          * @return void
243          */
244         function SetFckConfigUrl($url){
245             $this->   fck_configUrl = $url;
246         }
247
248         /**
249          * Takes in a standard single row from $this->form information and
250          * generated and html form input string
251          *
252          * @param String $input string eg.
State:10:select:ACT|ACT,NSW|NSW,VIC|VIC,QLD|QLD,SA|SA,NT|NT,WA|WA,TAS|TAS:::optional:notype
253          * @return Html form input
254          */
255         function inputfield($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;-
1\"     "     ;
277             }else{
278                 @$valid_str = "     validitycheck=\"     $field[1];$field[0];optional;notype;0;-
1\"     "     ;
279             }
280             if (!isset($field[3])){
281                 $field[3]=''; // if value not set then create and set to blank
282             }
283             //nulled Error as value can be blank in this case
284             if (@isset($field[4])){
285
286                 $func = explode(',', trim($field[4]));
287
288                 foreach($func AS $value){
289                     if(preg_match( "/^J\|/"          , $value)){
290                         $value=str_replace("J|"          ,'',$value);
291                         $func_script = trim($value)." "          ;
292                     }
293                     if(preg_match( "/^C\|/"          , $value)){
294                         $value=str_replace("C|"          ,'',$value);
295                         $func_class = "class="          .trim($value)." "          ;
296                     }
297                     if(preg_match( "/^I\|/"          , $value)){
298                         $value=str_replace("I|"          ,'',$value);
299                         $func_id = "id="          .trim($value)." "          ;
300                     }
301                     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_LABLE"          ) $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=\"lable\">                    $field[0]</div>
<div class=\"field\">               "     .@$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

```
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',
'var4'=>'<strong>frank</strong>'});<br />
13     * <br />
14     * $ArrayVars[0]['name'] = 'john';<br />
15     * $ArrayVars[0]['age'] = '14';<br />
16     * $ArrayVars[1]['name'] = 'mary';<br />
17     * $ArrayVars[1]['age'] = '42';<br />
18     * $ArrayVars[2]['name'] = 'frank';<br />
19     * $ArrayVars[2]['age'] = '98';<br />
20     * <br />
21     * $template->assignRepeat('agelist', $ArrayVars);<br />
22     * $template->replace('../', '../../');<br />
23     * <br />
24     * $template->display();<br />
25     * <br />
26     *
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->   SEOurl){
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))? $name : NULL;
144         }
145
146         public function genterateDisplayTable($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             $tr = '';
156             $td = '';
157             $link = '';
158             $CSS_id='';
159             $cWidth = '';
160 
161             //return if nothing to build
162             /*if (count($content) == 0 || $content == false){
163                 return "None";
164             }*/
165             //cycle echo throught the records
166             if($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
create header
173                                 $countColumn = 0;
174                                 foreach($val AS $key=>   $value){
175 
176                                         //if(isset($this->row_class_field_name)){
177                                             if($key == $this->   row_class_field_name){
178                                                 $className = $value." "          .$this-
>   row_class_name;
179                                             }else{
180                                                 $className = $this->   row_class_name;
181                                             }
182 
183                                         //}
184 
185                                         //is there a need to reset the header names
186                                          if (is_array($this->   headerArray)){
187                                             //if current header is in headerArray change the display
header name
188                                             if(array_key_exists($key, $this->   headerArray)){
189                                                 $columnVal = $this->   getOrderType($this-
>   headerArray[$key], $key);
190                                             }else{
191                                                 $columnVal = $this->   getOrderType($key, $key); //just
add header, no change
192                                             }
193                                         }else{
194                                                 $columnVal = $this->   getOrderType($key, $key); //just
add header, no change
195                                         }
196 
197                                         //if the identifier used for the linking to a page, e.g primary
id
198                                         if($key == $this->   identifier){
199                                             $id = $val[$key];
200                                             $page = (isset($this->   identifier_link_page))? $this-
>   identifier_link_page : $_SERVER['PHP_SELF'] ;
201                                             if($this->   nolink){
202                                                 $link = "id="          .$id;
203                                             }else{
204                                                 if(isset($this->   link_field)){
205                                                     $action = $val[$this->   link_field];
206                                                 }elseif(isset($this->   link_action)){
207                                                     $action = $this->   link_action;
208                                                 }else{
209                                                     $action = "show"          ;
210                                                 }
211                                                 if($this->   SEOurl){
212                                                     $link =
"onclick=\"location.href='"             .$page."/"          .$action."/"          .$id."'\"
id="     .$id;
213                                                 }else{
214                                                     $action = "action="          .$action;
215                                                     $link =
"onclick=\"location.href='"             .$page."?"          .$action."&id="             .$id."'\&q
```

```
uot; id="     .$id;
                                        }
                                        //$link =
"onclick=\"location.href='".$page."?".$action."&id=".$id."'\&q
uot; id=".$id;
                                    }
                                }
                                if(!in_array($key, $this->   removeColumnArray)){
                                    if(!empty($this->   columnsWidth[$countColumn])){
                                        $cWidth = ' width="'     .$this-
>   columnsWidth[$countColumn].'"'     ;
                                    }
                                    $column
.="     <th   $cWidth>   "     .$columnVal."</th>"               ; // append to header
                                    $td .= $this->   buildTd($key, $value);
                                }

                                // is there a need to a filter row
                                if (is_array($this->   filterArray) || !$this->   rowsOnly){
                                    //if current header is in filterArray change then add the
filter to the filter row
                                    if(array_key_exists($key, $this->   filterArray)){
                                        $filterVal = $this->   getFilterType($this-
>   filterArray[$key], $key, $content2); // format filter type
                                    }
                                    if(!in_array($key, $this->   removeColumnArray)){
                                        if($countColumn == 0){
                                            $primaryid = (!empty($this->   id))?"<input
type='hidden' name='id' value='"     .$this->   id."'>"                :'';
                                            $filter .="<td
NOWRAP>"        .$primaryid.$filterVal."</td>"                ; // append then Field to filter Row
                                        }else{
                                            $filter .="<td
NOWRAP>"        .$filterVal."</td>"                ; // append then Field to filter Row
                                        }
                                        $countColumn++;
                                    }
                                }

                            }
                        }else{
                            //cycle thought each field in row

                            foreach($val AS $key=>   $value){
                                 if(!in_array($key, $this->   removeColumnArray)){
                                    $td .= $this->   buildTd($key, $value);
                                }

                                //if(isset($this->row_class_field_name)){
                                    if($key == $this->   row_class_field_name){
                                        $className = $value." "          .$this-
>   row_class_name;
                                    }else{
                                        $className = $this->   row_class_name;
                                    }

                                //}

                                //if the identifier used for the linking to a page, e.g primary
id
                                if($key == $this->   identifier){
                                    $id = $val[$key];
                                    $page = (isset($this->   identifier_link_page))? $this-
>   identifier_link_page : $_SERVER['PHP_SELF'] ;
                                    if($this->   nolink){
                                        $link = "id="          .$id;
                                    }else{
                                        if(isset($this->   link_field)){
                                            $action = "action="          .$val[$this-
>   link_field];
                                        }elseif(isset($this->   link_action)){
                                            $action = "action="          .$this->   link_action;
                                        }else{
                                            $action = "action=show"          ;
                                        }

                                        $link =
"onclick=\"location.href='"              .$page."?"          .$action."&id="              .$id."'\&q
uot; id="     .$id;
                                    }
```

```
280                                     }
281                                 }
282                             }
283                             $tr .= "<tr "             .$link ."
class=\""          .$className." row\">"                 .$td."</tr>\n"                ;// wrap in table
row  and append
284                             $td = '';
285                         }
286                 }
287                 $footer = (isset($this->   footer))? "<TFOOT>"                .$this-
>   footer."</TFOOT>"                   : '';
288                 unset($this->   footer);
289
290                 if($this->   rowsOnly){
291                     $table = $tr;
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><tr class=\"filter
noprint\">"             .$filter."</tr>\n\t"                .$tr."</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 'COMPLIED' :
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."\" >
320                         <option value=\"eq\" selected>=</option>
321                         <option value=\"greater\">>=</option>
322                         <option value=\"less\"><=</option>
323                         </SELECT>
324                         <input type=\"text\" name=\"flt_"                      .$key."\"
size=\"2\" />"                  ; break;
325                     case 'MONEY' :
326                         $field = "<SELECT name=\"dir_"                  .$key."\" >
327                         <option value=\"eq\" selected>=</option>
328                         <option value=\"greater\">>=</option>
329                         <option value=\"less\"><=</option>
330                         </SELECT>
331                         <input type=\"text\" name=\"flt_"                      .$key."\"
size=\"2\" />"                  ; break;
332                     default: $field = '';
333                 }
334
335                 return $field;
336             }
337
338
339             private function getOrderType($type, $key){
340
341                 $field = "<div
id=\""          .$key."\">"                  .$type."</div>"                ;
342
343                 return $field;
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
357           }else{
358               $value2 = str_replace(',', '',$value);
359               if(is_numeric($value2)){
360                   $td ="<td style=\"text-
align:center\">"             .$value."</td>"               ;
361               }else{
362                   $td ="<td>"               .$value."</td>"               ; // append then Field to
Row
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]

```php
<?php
/**
 * Tools Function,
 * <br />
 * At set of general tool to help with development<br />
 *
 *
 * @author Jason Stewart <jason@lexxcom.com.au>
 * @version 1.0
 * @package PeopleScope
 * @subpackage Base
 */


/**
 * Global style for error messages
 * @global string $GLOBALS['style']
 */
$GLOBALS['style'] = "style=\"font-family: Arial, Helvetica, sans-serif; font-size: 11px;
padding:10px; background-color: #ffcccc; width:100%; border:solid 1px #000\""          ;
/**
 * Global trace debugging
 * @global string $GLOBALS['trace']
 */
$GLOBALS['trace']=array();

/**
 * A debuging tool
 *
 * This will produce a error message to the screen displaying the value enter
 * it will take in vars | arrays | object
 * it will show the page | Line | function that the echo accured
 * and can be turned off thought the constant TURN_ON_PP
 *
 * @param Mixed $var Value that will be echoed out
 * @param bool $tracer true display to screen, false store in global $trace
 *  return void
 */
Function pp($var, $tracer=false){
    if (TURN_ON_PP){
        global $style;
        $info = debug_backtrace();
        $line =  trim(str_replace($_SERVER['DOCUMENT_ROOT'], '', $info[0]['file'])).
: "     .$info[0]['line']." -- Function : "          .@$info[1]['function'];

        if(is_array($var) || is_object($var)){
            $responce = "     <div    $style><h2>Notification
message</h2><h3>           "     .$line."</h3>\n<pre>"                    .html_entity_decode(
print_r($var,true))."</pre></div>"                    ;
        }else{
            $responce = "     <div    $style><h2>Notification
message</h2><h3>           "     .$line."     </h3>\n<pre>           $var</pre></div>           "     ;
        }

        if($tracer){
            global $trace;
            $trace[] = $responce;
        }else{
            echo $responce;
        }
    }
}

/**
 * Debuging tool that that will store the outcomes into a $GLOBAL['trace'] var
 *
 * @global Array $GLOBAL['trace']
```

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

```
140
141         $line .= "<h4>"                 .$name."</h4><blockquote><br/>"                             ;
142         if (isset($array)){
143             foreach($array AS $key=>    $value){
144                 $line .= "['"           .$key."']=   "                       .
print_r($value,1)."<br />"                  ;
145             }
146         }
147         $line .= "</blockquote>"                ;
148         return $line;
149     }
150
151
152     /**
153      * Will take in a String and a start and end delimiter and will return the content between the
delimiter
154      *
155      * @param String $string String to parse
156      * @param String $beg_tag start delimiter
157      * @param String $close_tag end delimiter
158      * @param Bool $remove_tag true return content without delimiter
159      * @return Array all content found between delimiters
160      */
161     function parseArray($string, $beg_tag, $close_tag, $remove_tag=true){
162         $on = preg_match_all("      ($beg_tag(.*)$close_tag)siU"      , $string, $matching_data);
163
164         if($remove_tag){
165             return $matching_data[1][0];
166         }
167
168         return $matching_data[0][0];
169     }
170
171     /**
172      * Will take in a list of accociated array names and remove
173      * from that array via referance
174      *
175      * @param String $elements Comma seperated list of names
176      * @param Array $array The array to remove them from via referance
177      */
178     function stripElements($elements, &    $array){
179         $elementsArray = explode(',', $elements);
180
181         foreach($elementsArray AS $value){
182             unset($array[$value]);
183         }
184     }
185     /**
186      * This will take in a name for the field types and a date that you wish and format
187      * an option list for select boxes, and return them in an associated array for year, month and day
188      *
189      * @param String $name Name used for selection box
190      * @param String $date Date That should be selected
191      * @param Integer $yearRange How many years should be displayed
192      * @param Integer $startyear What year should the list start at
193      *
194      * @return an Array of day,month,Year holding the required option list for select box
195      */
196     function createDateField($date, $yearRange = 10, $startyear = NULL){
197
198         if (!isset($startyear)){
199             $startyear = date('Y');
200         }
201
202         //lets do some cleaning up on the date format to make it easy to work with.
203         $date = trim($date);
204         $date = str_replace(array('\\','/'), '-', $date);
205         $date = str_replace(' ', '', $date);
206
207         list($day, $month, $year) = explode('-', $date);
208
209         $months = array(""          ,"January"          , "February"          , "March"          ,
"April"          , "May"          , "June"          , "July"          , "August"          ,
"September"          , "October"          , "November"          , "December"          );
210
211         for($i = 1; $i <    count($months) ; $i++){
212             $selectmonth = ($i == $month)? ' selected="selected"'          : '';
213             @$dateRet['month'] .= "     <option
value=\"     $i\"     $selectmonth>   "     .$months[$i]."</option>\n"               ;
214         }
```

```
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 accociated name type and remove the elements and replace the with a $name
field with a value of the compiled date,
247      * if none found then will put in todays date
248      * accociated name types :<br />
249      * $name.'__day'<br />
250      * $name.'__month'<br />
251      * $name.'__year'<br />
252      * <br />
253      * $name = $name.'__year-'$name.'__month-'$name.'__day 00::00::00' <br />
254      * <br />
255      * @param String $name the name ofthe fields we are looking for
256      * @param Array &$listArray     alternate array to look at, $_REQUEST by default
257      * @return String database date format
258      */
259     function formatDateResponce($name, &    $listArray=NULL){
260         if (!$listArray){
261             $listArray = $_REQUEST;
262         }
263
264         if(isset($listArray[$name.'__day'])){
265             $nameDay = $listArray[$name.'__day'];
266             unset($listArray[$name.'__day']);
267         }else{
268             $nameDay = date('j');
269         }
270
271         if(isset($listArray[$name.'__month'])){
272             $nameMonth = $listArray[$name.'__month'];
273             unset($listArray[$name.'__month']);
274         }else{
275             $nameMonth = date('n');
276         }
277
278         if(isset($listArray[$name.'__year'])){
279             $nameYear = $listArray[$name.'__year'];
280             unset($listArray[$name.'__year']);
281         }else{
282             $nameYear = date('Y');
283         }
284
285         $listArray[$name] = ($nameYear.'-'.$nameMonth.'-'.$nameDay.' 00:00:00' );
286
287         return $listArray[$name];
288     }
289
290     /**
291      * 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 referance
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            $etimestamp = mktime(0,0,0,$em,$ed,$ey);
321            if(empty($startdate)){
322                $ARRAY[$end] = date('Y-m-d 00:00:00', $etimestamp);
323                return;
324            }
325        }
326
327        if ($stimestamp <    $etimestamp){
328            $ARRAY[$start] = date('Y-m-d 00:00:00', $stimestamp);
329            $ARRAY[$end] = date('Y-m-d 00:00:00', $etimestamp);
330        }else{
331            $ARRAY[$end] = date('Y-m-d 00:00:00', $stimestamp);
332            $ARRAY[$start] = date('Y-m-d 00:00:00', $etimestamp);
333        }
334
335    }
336
337    /**
338     * converts a database date value and conversts to Au date format
339     * 2001-04-02 = 02/04/2001
340     * @param String $dbdate database date format yyyy-mm-dd
341     * @return String Date dd/mm/yyyy
342     */
343    function databaseToUI($dbdate){
344
345        list($dbdate) = explode(' ' , $dbdate);
346
347        list($year, $month, $day) = explode('-' , $dbdate);
348
349        return  $day ."/"          .$month."/"          .$year;
350    }
351
352    /**
353     * return file extention based on Mime type for common doc types
354     * doc, docx, pdf
355     *
356     * @param String $type Mime type
357     * @return String Document extention
358     */
359    function fileExt($type){
360        $ext = false;
361        switch($type){
362            case 'application/msword' : $ext = 'doc'; break;
363            case 'application/pdf' : $ext = 'pdf'; break;
364            case 'application/vnd.openxmlformats-officedocument.wordprocessingml.document' : $ext =
'docx'; break;
365        }
366        return $ext;
367    }
368    /**
369     * Will take in a percent amount and return values in mins
370     * eg. 50% = 30mins
```

```php
     * @param Integer $percent
     * @return Integer Minutes
     */
    function convertDecimalToMinutes($percent){
        list($hours, $minutes)= explode('.', number_format($percent,2));
        $minutes = ceil(($minutes*60)/100);
        $in_minutes=($hours*60)+$minutes;
        return $in_minutes;
    }

    /**
     * Will take in a amount in minutes and return the value in the amount of hours : minute
     *
     *  124 minutes = 2:04
     *
     * @param Integer $Minutes
     * @return String hh:mm
     */
    function convertMinutes2Hours($Minutes)
    {
        if ($Minutes <    0)
        {
            $Min = Abs($Minutes);
        }
        else
        {
            $Min = $Minutes;
        }
        $iHours = Floor($Min / 60);
        $Minutes = ($Min - ($iHours * 60)) / 100;
        $tHours = $iHours + $Minutes;
        if ($Minutes <    0)
        {
            $tHours = $tHours * (-1);
        }
        $aHours = explode("."          , $tHours);
        $iHours = $aHours[0];
        if (empty($aHours[1]))
        {
            $aHours[1] = "00"          ;
        }
        $Minutes = $aHours[1];
        if (strlen($Minutes) <    2)
        {
            $Minutes = $Minutes ."0"          ;
        }
        $tHours = $iHours ."."          . $Minutes;
        return $tHours;
    }

    function Box($content, $header, $discription){
        $html ='<div  class="tab">
                <div id="tab-header">
                    <h1>'      .$header.'</h1>
                </div>
                <div id="tab-text">
                    '.$discription.'
                </div>
                <div id="tab-body">
                    '.$content.'<br class="clear">
                </div>
                <div id="tab-foot"></div>
            </div>'      ;
        return $html;

    }

    function warning($content){
        $html ='<div class="warning"><img src="images/warning.png" />
                '.$content.'
            </div>'      ;
        return $html;

    }

    ?>
```

Package default

File Source for administration.class.php

Documentation for this file is available at administration.class.php

```
1     <?php
2
3     class administration {
4
5         private $db_connect;
6         private $db;
7         public $lastError;
8         public $table;
9         public $template;
10        private $error;
11        public $admin;
12        private $fields =array('administration_id', 'group_name', 'create_advert', 'edit_advert',
'remove_advert', 'create_template', 'edit_template', 'remove_template', 'add_user', 'edit_user',
'delete_user', 'edit_status', 'edit_referral');
13        private $fields_required = NULL;
14        private $fields_validation_type = array ('administration_id'=>   'INT',
'group_name'=>   'TEXT', 'create_advert'=>   'BOOL', 'edit_advert'=>   'BOOL',
'remove_advert'=>   'BOOL', 'create_template'=>   'BOOL', 'edit_template'=>   'BOOL',
'remove_template'=>   'BOOL', 'add_user'=>   'BOOL', 'edit_user'=>   'BOOL', 'delete_user'=>   'BOOL',
'edit_status'=>   'BOOL', 'edit_referral'=>   'BOOL');
15
16        function __construct(){
17            $this->   db = new db();
18
19            try {
20                $this->   db_connect = $this->   db->   dbh;
21            } catch (CustomException $e) {
22                $e->   logError();
23            }
24
25            $this->   table = new table();
26            $this->   template = new template();
27
28        }
29
30        Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
31
32            $sql = "SELECT administration_id,
33                        group_name,
34                        IF(create_advert=1, 'YES','NO') AS create_advert,
35                        IF(edit_advert=1, 'YES','NO') AS edit_advert,
36                        IF(remove_advert=1, 'YES','NO') AS remove_advert,
37                        IF(create_template=1, 'YES','NO') AS create_template,
38                        IF(edit_template=1, 'YES','NO') AS edit_template,
39                        IF(remove_template=1, 'YES','NO') AS remove_template,
40                        IF(add_user=1, 'YES','NO') AS add_user,
41                        IF(edit_user=1, 'YES','NO') AS edit_user,
42                        IF(delete_user=1, 'YES','NO') AS delete_user,
43                        IF(edit_status=1, 'YES','NO') AS edit_status,
44                        IF(edit_referral=1, 'YES','NO') AS edit_referral
45                    FROM administration"     ;
46
47            if(is_array($filter)){
48                  foreach($filter AS $key=>   $value){
49                      if ($value != 'NULL'  && !empty(        $value)){
50                          $sql .=  " AND "          . $value;
51                      }
52                  }
53            }
54
55            if($orderby){
56                  $sql .= " ORDER BY "          . $orderby." "          .$direction;
57            }
58
59            try{
60                 $result = $this->   db->   select($sql);
61            }catch(CustomException $e){
```

```
62                 echo $e->   queryError($sql);
63             }
64
65             return $result;
66         }
67
68         Private function create($source){
69                 try{
70                     $this->   db_connect->   beginTransaction();
71
72                     foreach($source['administration'] AS $key=>   $val){
73                         $field[] = $key;
74                         $value[] = ":"          .$key;
75                     }
76
77                     $sql = "INSERT INTO administration ("          .implode(', ',$field).") VALUES
("      .implode(', ',$value).");"          ;
78
79                     foreach($source['administration'] AS $key=>   $val){
80                         $exec[":"          .$key] = $val;
81                     }
82
83                     try{
84                         $pid = $this->   db->   insert($sql, $exec);
85                     }catch(CustomException $e){
86                         throw new CustomException($e->   queryError($sql));
87                     }
88
89                     $pid = $this->   db_connect->   lastInsertId();
90                 }
91
92                 catch (CustomException $e) {
93                     $e->   queryError($sql);
94                     $this->   db_connect->   rollBack();
95                     return false;
96                 }
97
98                 return $pid;
99         }
100
101
102        Private function read($id){
103
104            $sql = "SELECT administration_id,
105                        group_name,
106                        create_advert,
107                        edit_advert,
108                        remove_advert,
109                        create_template,
110                        edit_template,
111                        remove_template,
112                        add_user,
113                        edit_user,
114                        delete_user,
115                        edit_status,
116                        edit_referral
117                    FROM administration WHERE administration_id = "     . $id ."";
118
119
120                $stmt = $this->   db_connect->   prepare($sql);
121                $stmt->   execute();
122
123                try{
124                     $result = $this->   db->   select($sql);
125                }catch(CustomException $e){
126                     echo $e->   queryError($sql);
127                }
128
129                return $result[0];
130
131
132        }
133
134        Private function update($source, $id){
135                if($id <=    1){
136                    header('Location:administration.php?action=show&id='    .$id);
137                }
138                try{
139                    $this->   db_connect->   beginTransaction();
140
```

```
141                     foreach($source['administration'] AS $key=>   $val){
142                         $field[] = $key." = :"          .$key;
143                     }
144
145                     $sql = "UPDATE administration SET "          .implode(', ',$field)." WHERE
administration_id ="     . $id;
146
147                     $stmt = $this->   db_connect->   prepare($sql);
148
149                     foreach($source['administration'] AS $key=>   $val){
150                         $exec[":"          .$key] = $val;
151                     }
152                     $stmt->   execute($exec);
153
154
155                     if ($stmt->   errorCode() != 00000 )
156                     {
157                         $this->   error->   set_error($stmt->   errorInfo(), $sql);
158                         $this->   error->   get_error();
159                         $this->   db_connect->   rollBack();
160                     }
161
162
163                     //$id = $this->db_connect->lastInsertId();
164                     $source['administration']['administration_id'] = $id;
165
166                 }
167
168                 catch (Exception $e) {
169                     $this->   error->   set_error($stmt->   errorInfo(), $sql);
170                     $this->   error->   get_error();
171                     $this->   db_connect->   rollBack();
172                 }
173
174                 header('Location:administration.php?action=show&id='    .$id);
175
176         }
177
178         Private function remove($id){
179                 if(empty($id)){
180                     return false;
181                 }
182
183                 $sql = "UPDATE administration SET delete_date=NOW() WHERE administration_id
="     . $id;
184
185                 try{
186                     $result = $this->   db->   update($sql);
187                 }catch(CustomException $e){
188                     echo $e->   queryError($sql);
189                 }
190                 return true;
191         }
192
193
194         /****************** END CRUD METHOD************************/
195
196         public function getAdministrationList($type='TABLE',$orderby=NULL, $direction='ASC',
$filter=NULL){
197
198             $result = $this->   lists($orderby, $direction, $filter);
199
200             switch(strtoupper($type)){
201
202                 case 'AJAX' : $this->   table->   setRowsOnly();
203                               $this->   table->   removeColumn(array('administration_id'));
204                               $this->   table->   setIdentifier('administration_id');
205                               $this->   table->   setIdentifierPage('administration');
206                               echo $this->   table->   genterateDisplayTable($result);
207
208                     BREAK;
209                 case 'TABLE' :
210                 DEFAULT :
211                     $this->   table->   setHeader(array(
212                             'administration_id'=>   'Administration Id',
213                             'group_name'=>   'Group Name',
214                             'create_advert'=>   'Create Advert',
215                             'edit_advert'=>   'Edit Advert',
216                             'remove_advert'=>   'Remove Advert',
217                             'create_template'=>   'Create Template',
```

```
                        'edit_template'=>   'Edit Template',
                        'remove_template'=>   'Remove Template',
                        'add_user'=>   'Add User',
                        'edit_user'=>   'Edit User',
                        'delete_user'=>   'Delete User',
                        'edit_status'=>   'Edit Status',
                        'edit_referral'=>   'Edit Referral'));

                $this->   table->   setFilter(array(
                        'administration_id'=>   'TEXT',
                        'group_name'=>   'TEXT',
                        'create_advert'=>   'TEXT',
                        'edit_advert'=>   'TEXT',
                        'remove_advert'=>   'TEXT',
                        'create_template'=>   'TEXT',
                        'edit_template'=>   'TEXT',
                        'remove_template'=>   'TEXT',
                        'add_user'=>   'TEXT',
                        'edit_user'=>   'TEXT',
                        'delete_user'=>   'TEXT',
                        'edit_status'=>   'TEXT',
                        'edit_referral'=>   'TEXT'));

                $this->   table->   removeColumn(array('administration_id'));

                $this->   table->   setIdentifier('administration_id');

                $this->   template->   content(BOX($this->   table-
>   genterateDisplayTable($result), 'List Administration Groups', 'This list shows the type of
administration Groups that are availaible, A users must always be assigned to a admin group to designate
there required access level for duty within the web site, you can create or change any of the group to
custimse your access levels but adminstration group can never be changed '));

                $this->   template->   display();
            }
        }

        Public function showAdministrationDetails($id, $return=false){
            $staffMember = $this->   read($id);

            $this->   template->   page('administration.tpl.html');

            $this->   templateAdministrationLayout($staffMember);

            if($id >    1){
                $this->   template->   assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='administration.php?action=edit&id="             .$id."'\">Edit</
div>"        );
            }

            echo $this->   template->   fetch();
        }

        Public function editAdministrationDetails($id){

            $staffMember = $this->   read($id);

            $name = 'editadministration';

            $this->   template->   page('administration.tpl.html');
            $this->   template->   assign('FORM-HEADER', '<form
action="administration.php?action=update&id='         .$id.'" method="POST"
name="'     .$name.'">'        );

            $this->   templateAdministrationLayout($staffMember, true);


            $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='administration.php?action=show&id=         "     .$id."'\">Cancel&lt
;/div>"        );


            $this->   template->   display();
        }


        Public function updateAdministrationDetails($id){
            if ($this->   Validate($_REQUEST)){
```

```
286
287                     $request = $_REQUEST;
288                     $table = 'administration';
289
290                     $save[$table]['group_name'] = $request['group_name'];
$save[$table]['create_advert'] = (isset($request['create_advert']))? 1 : 0 ;
$save[$table]['edit_advert'] = (isset($request['edit_advert']))? 1 : 0 ;
$save[$table]['remove_advert'] = (isset($request['remove_advert']))? 1 : 0 ;
$save[$table]['create_template'] = (isset($request['create_template']))? 1 : 0 ;
$save[$table]['edit_template'] = (isset($request['edit_template']))? 1 : 0 ;
$save[$table]['remove_template'] = (isset($request['remove_template']))? 1 : 0 ;
$save[$table]['add_user'] = (isset($request['add_user']))? 1 : 0 ;
$save[$table]['edit_user'] = (isset($request['edit_user']))? 1 : 0 ;
$save[$table]['delete_user'] = (isset($request['delete_user']))? 1 : 0 ;
$save[$table]['edit_status'] = (isset($request['edit_status']))? 1 : 0 ;
$save[$table]['edit_referral'] = (isset($request['edit_referral']))? 1 : 0 ;
291                     $save[$table]['modify_date'] = date('Y-m-d h:i:s');
292
293                     $this->   update($save, $id );
294
295                 }else{
296
297                     $staffMember = $this->   valid_field;
298                     $error = $this->   validation_error;
299
300                     $name = 'editgrant';
301
302                     $this->   template->   page('administration.tpl.html');
303
304                     foreach($validfields AS $value){
305                         if(isset($error[$value])){
306                             $this->   template->   assign('err_'.$value, "<span
class=\"error\">"                 .implode(',', $error[$value])."</spam>"                 );
307                         }
308                     }
309
310                     $this->   template->   assign('FORM-HEADER', '<form
action="administration.php?action=update&id='          .$id.'" method="POST"
name="'     .$name.'">'         );
311
312                     $this->   templateAdministrationLayout($staffMember, true);
313
314                     if($id >    1){
315                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.                $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='administration.php?action=show&id=          "     .$id."'\">Cancel&lt
;/div>"         );
316                     }
317                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
318
319                     $this->   template->   display();
320                 }
321             }
322
323
324             Public function createAdministrationDetails(){
325
326                 $name = 'createAdmin';
327
328                 $this->   template->   page('administration.tpl.html');
329                 $this->   template->   assign('FORM-HEADER', '<form
action="administration.php?action=save" method="POST"
name="'     .$name.'">'         );
330
331                 $this->   templateAdministrationLayout('', true);
332
333                 $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Save</div><div
class=\"button\"
onclick=\"location.href='administration.php?action=list'\">Cancel</div>                    "     );
334
335
336                 $this->   template->   display();
337             }
338
339             Public function saveAdministrationDetails(){
340
341                 if ($this->   Validate($_REQUEST)){
342
```

```
343                     $request = $_REQUEST;
344                     $table = 'administration';
345
346                     $save[$table]['group_name'] = $request['group_name'];$save[$table]['create_advert']
= $request['create_advert'];$save[$table]['edit_advert'] =
$request['edit_advert'];$save[$table]['remove_advert'] =
$request['remove_advert'];$save[$table]['create_template'] =
$request['create_template'];$save[$table]['edit_template'] =
$request['edit_template'];$save[$table]['remove_template'] =
$request['remove_template'];$save[$table]['add_user'] = $request['add_user'];$save[$table]['edit_user'] =
$request['edit_user'];$save[$table]['delete_user'] = $request['delete_user'];$save[$table]['edit_status']
= $request['edit_status'];$save[$table]['edit_referral'] = $request['edit_referral'];
347                     $save[$table]['create_date'] = date('Y-m-d h:i:s');
348
349                     $id = $this->   create($save);
350                     header('Location: administration.php?action=show&id='    .$id);
351
352                 }else{
353
354                     $staffMember = $this->   valid_field;
355
356                     $error = $this->   validation_error;
357
358                     $name = 'createadministration';
359
360                     $this->   template->   page('administration.tpl.html');
361
362                     foreach($error AS $key=>   $value){
363                         $this->   template->   assign('err_'.$key, "<span
class=\"error\">"                    .@implode(',', $error[12])."</spam>"                );
364                     }
365
366                     $this->   template->   assign('FORM-HEADER', '<form
action="administration.php?action=update&id='          .$id.'" method="POST"
name="'     .$name.'">'        );
367
368                     $this->   templateAdministrationLayout($staffMember, true);
369
370                     if($this->   admin->   checkAdminLevel(1)){
371                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.              $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='administration.php?action=show&id=         "     .$id."'\">Cancel&lt
;/div>"        );
372                     }
373                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
374
375                     $this->   template->   display();
376                 }
377             }
378
379             Public function deleteAdministrationDetails($id){
380                 $this->   remove($id);
381                 header('Location: administration.php');
382             }
383
384             private function templateAdministrationLayout($staffMember, $input = false, $inputArray=array()
){
385
386                     $id = $staffMember['administration_id'];
387
388                     if($id == 1){
389                         $this->   template->   assign('admin_warning',warning('This administrtion level
can not be edited'));
390                     }
391
392                     @$this->   template->   assign('group_title', $staffMember['group_name']);
393                     @$this->   template->   assign('administration_id', ($input)? $this->   template-
>   input('text', 'administration_id',
$staffMember['administration_id']):$staffMember['administration_id']);
394                     @$this->   template->   assign('group_name', ($input)? $this->   template-
>   input('text', 'group_name', $staffMember['group_name']):$staffMember['group_name']);
395                     @$this->   template->   assign('create_advert', ($input)? $this->   template-
>   input('checkbox', 'create_advert', $this->   template-
>   formatBoolean($staffMember['create_advert'])):$this->   template-
>   formatBoolean($staffMember['create_advert'] ));
396                     @$this->   template->   assign('edit_advert', ($input)? $this->   template-
>   input('checkbox', 'edit_advert', $this->   template-
>   formatBoolean($staffMember['edit_advert'])):$this->   template-
>   formatBoolean($staffMember['edit_advert'] ));
```

```
397                 @$this->   template->   assign('remove_advert', ($input)? $this->   template-
>   input('checkbox', 'remove_advert', $this->   template-
>   formatBoolean($staffMember['remove_advert'])):$this->   template-
>   formatBoolean($staffMember['remove_advert'] ));
398                 @$this->   template->   assign('create_template', ($input)? $this->   template-
>   input('checkbox', 'create_template', $this->   template-
>   formatBoolean($staffMember['create_template'])):$this->   template-
>   formatBoolean($staffMember['create_template'] ));
399                 @$this->   template->   assign('edit_template', ($input)? $this->   template-
>   input('checkbox', 'edit_template', $this->   template-
>   formatBoolean($staffMember['edit_template'])):$this->   template-
>   formatBoolean($staffMember['edit_template'] ));
400                 @$this->   template->   assign('remove_template', ($input)? $this->   template-
>   input('checkbox', 'remove_template', $this->   template-
>   formatBoolean($staffMember['remove_template'])):$this->   template-
>   formatBoolean($staffMember['remove_template'] ));
401                 @$this->   template->   assign('add_user', ($input)? $this->   template-
>   input('checkbox', 'add_user', $this->   template->   formatBoolean($staffMember['add_user'])):$this-
>   template->   formatBoolean($staffMember['add_user'] ));
402                 @$this->   template->   assign('edit_user', ($input)? $this->   template-
>   input('checkbox', 'edit_user', $this->   template->   formatBoolean($staffMember['edit_user'])):$this-
>   template->   formatBoolean($staffMember['edit_user'] ));
403                 @$this->   template->   assign('delete_user', ($input)? $this->   template-
>   input('checkbox', 'delete_user', $this->   template-
>   formatBoolean($staffMember['delete_user'])):$this->   template-
>   formatBoolean($staffMember['delete_user'] ));
404                 @$this->   template->   assign('edit_status', ($input)? $this->   template-
>   input('checkbox', 'edit_status', $this->   template-
>   formatBoolean($staffMember['edit_status'])):$this->   template-
>   formatBoolean($staffMember['edit_status'] ));
405                 @$this->   template->   assign('edit_referral', ($input)? $this->   template-
>   input('checkbox', 'edit_referral', $this->   template-
>   formatBoolean($staffMember['edit_referral'])):$this->   template-
>   formatBoolean($staffMember['edit_referral'] ));
406
407
408                 /*if(isset($id)){
409                     $this->template->assign('COMMENTS', $this->comment-
>getCommentBox($id, 'administration'));
410                 }*/
411
412       }
413
414       public function Validate($request){
415
416           unset($this->   valid_field);
417           unset($this->   validation_error);
418           $isvalid = True;
419
420           $validfields = $this->   fields;
421           $requiredfields = $this->   fields_required;
422           $fieldsvalidationtype = $this->   fields_validation_type;
423
424           foreach ($request AS $key=>   $value){ //lets strip put unwanted or security violation
fields
425               if(in_array($key, $validfields)){
426                   $this->   valid_field[$key] = $value; //pure fields
427               }
428           }
429
430           foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxs
431               if(!isset($this->   valid_field[$value])){
432                   $this->   valid_field[$value] = '';
433               }
434           }
435
436           if(count($requiredfields) >    0 ){
437               foreach($requiredfields AS $value){ // lets check all the required fields have a value
438                   if (empty($this->   valid_field[$value]) || $this->   valid_field[$value] ==
'NULL'){
439                       $this->   validation_error[$value][] = 'Field is Required'; //error field
440                       $isvalid = false;
441                   }
442               }
443           }
444
445
446
447           //now lets validate
```

```
448           foreach ($this->   valid_field AS $key=>   $value){
449               $value = trim($value);
450               if(!empty($value)){ // don't cheak if empty, alread done in required check
451
452                   switch(@$fieldsvalidationtype[$key]){
453                       case 'TEXTAREA': if (strlen($value) >    1024) {
454                                           $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
455                                           $isvalid = false;
456                                       } break;
457                       case 'TEXT': if (strlen($value) >    1024) {
458                                           $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
459                                           $isvalid = false;
460                                       } break;
461                       case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
462                                           $this->   validation_error[$key][] = 'not a valid SAP number';
463                                           $isvalid = false;
464                                       } break;
465                       case 'DECIMAL': if (!is_numeric($value)) {
466                                           $this->   validation_error[$key][] = 'Decimal value expected';
467                                           $isvalid = false;
468                                       } break;
469                       case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"          ) && (        $value != 1)) {
470                                           $this->   validation_error[$key][] = 'Please check';
471                                           $isvalid = false;
472                                       } break;
473                       case 'INT': if (!is_numeric($value) &&         $value != 'NULL' ){
474                                           $this->   validation_error[$key][] = 'Numeric value expected';
475                                           $isvalid = false;
476                                       } break;
477                       case 'DATE': $date = str_replace('/', '-', $value);
478                                    $date = str_replace("\\"          , '-', $date);
479                                       @list($day, $month, $year) = explode('-', $date);
480                                       if(!checkdate($month,$day, $year)){
481                                           $this->   validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
482                                           $isvalid = false;
483                                       } break;
484                       case 'YEAR':  if(!checkYear($value)){
485                                           $this->   validation_error[$key][] = 'incorrect year
format, expecting yyyy';
486                                           $isvalid = false;
487                                     } break;
488
489                   }
490               }
491           }
492
493           return $isvalid;
494
495       }
496
497       public function getSelectListOfAdministration($id, $selectBox=NULL){
498
499               if($selectBox == false){
500                   return $this->   template->   getListTable('administration', $id,
'administration_id', 'group_name');
501               }else{
502                   $select = "<select name=\"administration_id\">"                          ;
503                   $select .= "<option value=\"NULL\"></option>"                               ;
504                   $select .= $this->   template->   getListTable('administration', $id,
'administration_id', 'group_name', $selectBox);
505                   $select .= "</select>"              ;
506                   return $select;
507               }
508       }
509   }
```

File Source for advertisment.class.php

Documentation for this file is available at advertisment.class.php

```
1     <?php
2
3     class advertisment {
4
5         private $db_connect;
6         private $db;
7         public $lastError;
8         public $table;
9         public $template;
10        private $error;
11        public $admin;
12        private $fields =array('advertisment_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', 'create_date',
'create_by', 'modify_date', 'modify_by', 'delete_date', 'delete_by', 'question_id', 'tracking_id');
13        private $fields_required =array('title');
14        private $fields_validation_type = array ('advertisment_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'=>   'INT',
'create_date'=>   'TEXT', 'create_by'=>   'INT', 'modify_date'=>   'TEXT', 'modify_by'=>   'INT',
'delete_date'=>   'TEXT', 'delete_by'=>   'INT', 'question_id'=>   'INT', 'tracking_id'=>   'INT');
15
16        function __construct(){
17            $this->   db = new db();
18
19            try {
20                $this->   db_connect = $this->   db->   dbh;
21            } catch (CustomException $e) {
22                $e->   logError();
23            }
24
25            $this->   table = new table();
26            $this->   template = new template();
27
28        }
29
30        Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
31
32            $sql = "SELECT advertisment_id,
33                        title,
34                        catagory_id,
35                        template_id,
36                        office_id,
37                        dept_id,
38                        role_id,
39                        state_id,
40                        store_location_id,
41                        storerole_id,
42                        start_date,
43                        end_date,
44                        discription,
45                        requirments,
46                        upload_resume,
47                        cover_letter,
48                        status,
49                        employmenttype,
50                        create_date,
51                        create_by,
52                        modify_date,
53                        modify_by,
54                        delete_date,
55                        delete_by,
56                        question_id,
57                        tracking_id FROM advertisment WHERE (delete_date ='00-00-0000 00:00:00' OR
delete_date IS NULL)"     ;
```

```
58
59              if(is_array($filter)){
60                    foreach($filter AS $key=>   $value){
61                        if ($value != 'NULL'  && !empty(        $value)){
62                            $sql .=  " AND "          . $value;
63                        }
64                    }
65              }
66
67              if($orderby){
68                    $sql .= " ORDER BY "          . $orderby." "          .$direction;
69              }
70
71              try{
72                   $result = $this->   db->   select($sql);
73              }catch(CustomException $e){
74                   echo $e->   queryError($sql);
75              }
76
77              return $result;
78          }
79
80          Private function create($source){
81                  try{
82                      $this->   db_connect->   beginTransaction();
83
84                      foreach($source['advertisment'] AS $key=>   $val){
85                          $field[] = $key;
86                          $value[] = ":"          .$key;
87                      }
88
89                      $sql = "INSERT INTO advertisment ("          .implode(', ',$field).") VALUES
(" .implode(', ',$value).");"          ;
90
91                      foreach($source['advertisment'] AS $key=>   $val){
92                          $exec[":"          .$key] = $val;
93                      }
94
95                      try{
96                          $pid = $this->   db->   insert($sql, $exec);
97                      }catch(CustomException $e){
98                          throw new CustomException($e->   queryError($sql));
99                      }
100                     $this->   db_connect->   commit();
101                     $pid = $this->   db_connect->   lastInsertId();
102                 }
103
104                 catch (CustomException $e) {
105                     $e->   queryError($sql);
106                     $this->   db_connect->   rollBack();
107                     return false;
108                 }
109
110                 return $pid;
111         }
112
113
114         Private function read($id){
115
116             $sql = "SELECT advertisment_id,
117                     title,
118                     catagory_id,
119                     template_id,
120                     office_id,
121                     dept_id,
122                     role_id,
123                     state_id,
124                     store_location_id,
125                     storerole_id,
126                     start_date,
127                     end_date,
128                     discription,
129                     requirments,
130                     upload_resume,
131                     cover_letter,
132                     status,
133                     employmenttype,
134                     create_date,
135                     create_by,
136                     modify_date,
```

```
137                     modify_by,
138                     delete_date,
139                     delete_by,
140                     question_id,
141                     tracking_id FROM advertisment WHERE advertisment_id = "     . $id ." AND
(delete_date ='00-00-0000 00:00:00' OR delete_date IS NULL)"      ;
142
143
144                 $stmt = $this->   db_connect->   prepare($sql);
145                 $stmt->   execute();
146
147                 try{
148                      $result = $this->   db->   select($sql);
149                 }catch(CustomException $e){
150                      echo $e->   queryError($sql);
151                 }
152
153                 return $result[0];
154
155
156         }
157
158         Private function update($source, $id){
159                 try{
160                     $this->   db_connect->   beginTransaction();
161
162                     foreach($source['advertisment'] AS $key=>   $val){
163                         $field[] = $key." = :"          .$key;
164                     }
165
166                     $sql = "UPDATE advertisment SET "          .implode(', ',$field)." WHERE
advertisment_id ="      . $id;
167
168                     foreach($source['advertisment'] AS $key=>   $val){
169                         $exec[":"          .$key] = $val;
170                     }
171
172                     try{
173                             $pid = $this->   db->   update($sql, $exec);
174                     }catch(CustomException $e){
175                             throw new CustomException($e->   queryError($sql));
176                     }
177                     $this->   db_connect->   commit();
178                     $pid = $this->   db_connect->   lastInsertId();
179                 }
180
181                 catch (CustomException $e) {
182                     $e->   queryError($sql);
183                     $this->   db_connect->   rollBack();
184                     return false;
185                 }
186
187                 header('Location:advertisment.php?action=show&id='     .$id);
188
189         }
190
191         Private function remove($id){
192                 if(empty($id)){
193                     return false;
194                 }
195
196                 $sql = "UPDATE advertisment SET delete_date=NOW() WHERE advertisment_id ="          .
$id;
197
198                 try{
199                     $result = $this->   db->   update($sql);
200                 }catch(CustomException $e){
201                     echo $e->   queryError($sql);
202                 }
203                 return true;
204         }
205
206
207         /****************** END CRUD METHOD************************/
208
209         public function getAdvertismentList($type='TABLE',$orderby=NULL, $direction='ASC',
$filter=NULL){
210
211             $result = $this->   lists($orderby, $direction, $filter);
212
```

```
213           switch(strtoupper($type)){
214
215               case 'AJAX' : $this->   table->   setRowsOnly();
216                             $this->   table->   removeColumn(array('advertisment_id'));
217                             $this->   table->   setIdentifier('advertisment_id');
218                             $this->   table->   setIdentifierPage('advertisment');
219                             echo $this->   table->   genterateDisplayTable($result);
220
221                   BREAK;
222               case 'TABLE' :
223               DEFAULT :
224                   $this->   table->   setHeader(array(
225                                 'advertisment_id'=>   'Advertisment Id',
226   'title'=>   'Title',
227   'catagory_id'=>   'Catagory Id',
228   'template_id'=>   'Template Id',
229   'office_id'=>   'Office Id',
230   'dept_id'=>   'Dept Id',
231   'role_id'=>   'Role Id',
232   'state_id'=>   'State Id',
233   'store_location_id'=>   'Store Location Id',
234   'storerole_id'=>   'Storerole Id',
235   'start_date'=>   'Start Date',
236   'end_date'=>   'End Date',
237   'discription'=>   'Discription',
238   'requirments'=>   'Requirments',
239   'upload_resume'=>   'Upload Resume',
240   'cover_letter'=>   'Cover Letter',
241   'status'=>   'Status',
242   'employmenttype'=>   'Employmenttype',
243   'create_date'=>   'Create Date',
244   'create_by'=>   'Create By',
245   'modify_date'=>   'Modify Date',
246   'modify_by'=>   'Modify By',
247   'delete_date'=>   'Delete Date',
248   'delete_by'=>   'Delete By',
249   'question_id'=>   'Question Id',
250   'tracking_id'=>   'Tracking Id'));
251
252                   $this->   table->   setFilter(array(
253                                 'advertisment_id'=>   'TEXT',
254   'title'=>   'TEXT',
255   'catagory_id'=>   'TEXT',
256   'template_id'=>   'TEXT',
257   'office_id'=>   'TEXT',
258   'dept_id'=>   'TEXT',
259   'role_id'=>   'TEXT',
260   'state_id'=>   'TEXT',
261   'store_location_id'=>   'TEXT',
262   'storerole_id'=>   'TEXT',
263   'start_date'=>   'TEXT',
264   'end_date'=>   'TEXT',
265   'discription'=>   'TEXT',
266   'requirments'=>   'TEXT',
267   'upload_resume'=>   'TEXT',
268   'cover_letter'=>   'TEXT',
269   'status'=>   'TEXT',
270   'employmenttype'=>   'TEXT',
271   'create_date'=>   'TEXT',
272   'create_by'=>   'TEXT',
273   'modify_date'=>   'TEXT',
274   'modify_by'=>   'TEXT',
275   'delete_date'=>   'TEXT',
276   'delete_by'=>   'TEXT',
277   'question_id'=>   'TEXT',
278   'tracking_id'=>   'TEXT'));
279
280                   $this->   table->   removeColumn(array('advertisment_id'));
281
282                   $this->   table->   setIdentifier('advertisment_id');
283
284                   $this->   template->   content(Box($this->   table-
>   genterateDisplayTable($result),'Advertisment List', 'Shows the current listings for the
Advertisment'));
285
286                   $this->   template->   display();
287           }
288       }
289
290       Public function showAdvertismentDetails($id, $return=false){
```

```
291             $staffMember = $this->   read($id);
292
293             $this->   template->   page('advertisment.tpl.html');
294
295             $this->   templateAdvertismentLayout($staffMember);
296
297             //if($this->checkAdminLevel(1)){
298                 $this->   template->   assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='advertisment.php?action=edit&id="              .$id."'\">Edit</di
v>"         );
299             //}
300
301             echo $this->   template->   fetch();
302         }
303
304         Public function editAdvertismentDetails($id){
305
306             $staffMember = $this->   read($id);
307
308             $name = 'editadvertisment';
309
310             $this->   template->   page('advertisment.tpl.html');
311             $this->   template->   assign('FORM-HEADER', '<form
action="advertisment.php?action=update&id='          .$id.'" method="POST"
name="'      .$name.'">'          );
312
313             $this->   templateAdvertismentLayout($staffMember, true);
314
315             $this->   template->   assign('FUNCTION', "      <div class=\"button\"
onclick=\"document.      $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='advertisment.php?action=show&id=          "      .$id."'\">Cancel</
div>"         );
316
317             $this->   template->   display();
318         }
319
320
321         Public function updateAdvertismentDetails($id){
322
323             if ($this->   Validate($_REQUEST)){
324
325                     $request = $_REQUEST;
326                     $table = 'advertisment';
327
328                     $save[$table]['title'] = $request['title'];
329                     $save[$table]['catagory_id'] = $request['catagory_id'];
330                     $save[$table]['template_id'] = $request['template_id'];
331                     $save[$table]['office_id'] = $request['office_id'];
332                     $save[$table]['dept_id'] = $request['dept_id'];
333                     $save[$table]['role_id'] = $request['role_id'];
334                     $save[$table]['state_id'] = $request['state_id'];
335                     $save[$table]['store_location_id'] = $request['store_location_id'];
336                     $save[$table]['storerole_id'] = $request['storerole_id'];
337                     $save[$table]['start_date'] = formatDateUI($request['start_date']);
338                     $save[$table]['end_date'] = formatDateUI($request['end_date']);
339                     $save[$table]['discription'] = $request['discription'];
340                     $save[$table]['requirments'] = $request['requirments'];
341                     $save[$table]['upload_resume'] = $request['upload_resume'];
342                     $save[$table]['cover_letter'] = $request['cover_letter'];
343                     $save[$table]['status'] = $request['status'];
344                     $save[$table]['employmenttype'] = $request['employmenttype'];
345                     $save[$table]['create_by'] = $request['create_by'];
346                     $save[$table]['modify_by'] = $request['modify_by'];
347                     $save[$table]['delete_by'] = $request['delete_by'];
348                     $save[$table]['question_id'] = $request['question_id'];
349                     $save[$table]['tracking_id'] = $request['tracking_id'];
350
351                     $save[$table]['modify_date'] = date('Y-m-d h:i:s');
352
353                     $this->   update($save, $id );
354
355                 }else{
356
357                     $staffMember = $this->   valid_field;
358                     $error = $this->   validation_error;
359
360                     $name = 'editgrant';
361
362                     $this->   template->   page('advertisment.tpl.html');
```

```
363
364                     foreach($error AS $key=>   $value){
365                         $this->   template->   assign('err_'.$key, "<span
class=\"error\">"                     .@implode(',', $error[$key])."</spam>"                );
366                     }
367
368                     $this->   template->   assign('FORM-HEADER', '<form
action="advertisment.php?action=update&id='          .$id.'" method="POST"
name="'     .$name.'">'        );
369
370                     $this->   templateAdvertismentLayout($staffMember, true);
371
372                     //if($this->admin->checkAdminLevel(1)){
373                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.                $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='advertisment.php?action=show&id=         "     .$id."'\">Cancel</
div>"        );
374                     //}
375                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
376
377                     $this->   template->   display();
378                 }
379             }
380
381
382             Public function createAdvertismentDetails(){
383
384                 $name = 'createAdmin';
385
386                 $this->   template->   page('advertisment.tpl.html');
387                 $this->   template->   assign('FORM-HEADER', '<form
action="advertisment.php?action=save" method="POST" name="'                          .$name.'">'        );
388
389                 $this->   templateAdvertismentLayout('', true);
390
391                 $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Save</div><div
class=\"button\"
onclick=\"location.href='advertisment.php?action=list'\">Cancel</div>                    "     );
392
393
394                 $this->   template->   display();
395             }
396
397             Public function saveAdvertismentDetails(){
398
399                 if ($this->   Validate($_REQUEST)){
400
401                         $request = $_REQUEST;
402                         $table = 'advertisment';
403
404                         $save[$table]['title'] = $request['title'];
405                         $save[$table]['catagory_id'] = $request['catagory_id'];
406                         $save[$table]['template_id'] = $request['template_id'];
407                         $save[$table]['office_id'] = $request['office_id'];
408                         $save[$table]['dept_id'] = $request['dept_id'];
409                         $save[$table]['role_id'] = $request['role_id'];
410                         $save[$table]['state_id'] = $request['state_id'];
411                         $save[$table]['store_location_id'] = $request['store_location_id'];
412                         $save[$table]['storerole_id'] = $request['storerole_id'];
413                         $save[$table]['start_date'] = formatDateUI($request['start_date']);
414                         $save[$table]['end_date'] = formatDateUI($request['end_date']);
415                         $save[$table]['discription'] = $request['discription'];
416                         $save[$table]['requirments'] = $request['requirments'];
417                         $save[$table]['upload_resume'] = $request['upload_resume'];
418                         $save[$table]['cover_letter'] = $request['cover_letter'];
419                         $save[$table]['status'] = $request['status'];
420                         $save[$table]['employmenttype'] = $request['employmenttype'];
421                         $save[$table]['create_by'] = $request['create_by'];
422                         $save[$table]['modify_by'] = $request['modify_by'];
423                         $save[$table]['delete_by'] = $request['delete_by'];
424                         $save[$table]['question_id'] = $request['question_id'];
425                         $save[$table]['tracking_id'] = $request['tracking_id'];
426
427                         $save[$table]['create_date'] = date('Y-m-d h:i:s');
428
429                         $this->   create($save);
430
431                 }else{
```

```
432
433                     $staffMember = $this->   valid_field;
434
435                     $error = $this->   validation_error;
436
437                     $name = 'createadvertisment';
438
439                     $this->   template->   page('advertisment.tpl.html');
440
441                     foreach($error AS $key=>   $value){
442                         $this->   template->   assign('err_'.$key, "<span
class=\"error\">"                     .@implode(',', $value)."</spam>"                );
443                     }
444
445                     $this->   template->   assign('FORM-HEADER', '<form
action="advertisment.php?action=save" method="POST" name="'                              .$name.'">'        );
446
447                     $this->   templateAdvertismentLayout($staffMember, true);
448
449                     //if($this->admin->checkAdminLevel(1)){
450                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.                $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='advertisment.php\">Cancel</div>                    "     );
451                     //}
452                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
453
454                     $this->   template->   display();
455             }
456         }
457
458         Public function deleteAdvertismentDetails($id){
459             $this->   remove($id);
460             header('Location: advertisment.php');
461         }
462
463         private function templateAdvertismentLayout($staffMember, $input = false, $inputArray=array() ){
464
465                     $id = $staffMember['advertisment_id'];
466
467                     @$this->   template->   assign('advertisment_id', ($input)? $this->   template-
>   input('text', 'advertisment_id', $staffMember['advertisment_id']):$staffMember['advertisment_id']);
468                     @$this->   template->   assign('title', ($input)? $this->   template-
>   input('text', 'title', $staffMember['title']):$staffMember['title']);
469                     @$this->   template->   assign('catagory_id', ($input)? $this->   template-
>   input('text', 'catagory_id', $staffMember['catagory_id']):$staffMember['catagory_id']);
470                     @$this->   template->   assign('template_id', ($input)? $this->   template-
>   input('text', 'template_id', $staffMember['template_id']):$staffMember['template_id']);
471                     @$this->   template->   assign('office_id', ($input)? $this->   template-
>   input('text', 'office_id', $staffMember['office_id']):$staffMember['office_id']);
472                     @$this->   template->   assign('dept_id', ($input)? $this->   template-
>   input('text', 'dept_id', $staffMember['dept_id']):$staffMember['dept_id']);
473                     @$this->   template->   assign('role_id', ($input)? $this->   template-
>   input('text', 'role_id', $staffMember['role_id']):$staffMember['role_id']);
474                     @$this->   template->   assign('state_id', ($input)? $this->   template-
>   input('text', 'state_id', $staffMember['state_id']):$staffMember['state_id']);
475                     @$this->   template->   assign('store_location_id', ($input)? $this->   template-
>   input('text', 'store_location_id',
$staffMember['store_location_id']):$staffMember['store_location_id']);
476                     @$this->   template->   assign('storerole_id', ($input)? $this->   template-
>   input('text', 'storerole_id', $staffMember['storerole_id']):$staffMember['storerole_id']);
477                     @$this->   template->   assign('start_date', ($input)? $this->   template-
>   input('text', 'start_date', $staffMember['start_date']):$staffMember['start_date']);
478                     @$this->   template->   assign('end_date', ($input)? $this->   template-
>   input('text', 'end_date', $staffMember['end_date']):$staffMember['end_date']);
479                     @$this->   template->   assign('discription', ($input)? $this->   template-
>   input('text', 'discription', $staffMember['discription']):$staffMember['discription']);
480                     @$this->   template->   assign('requirments', ($input)? $this->   template-
>   input('text', 'requirments', $staffMember['requirments']):$staffMember['requirments']);
481                     @$this->   template->   assign('upload_resume', ($input)? $this->   template-
>   input('checkbox', 'upload_resume', $this->   template-
>   formatBoolean($staffMember['upload_resume'])):$this->   template-
>   formatBoolean($staffMember['upload_resume'] ));
482                     @$this->   template->   assign('cover_letter', ($input)? $this->   template-
>   input('checkbox', 'cover_letter', $this->   template-
>   formatBoolean($staffMember['cover_letter'])):$this->   template-
>   formatBoolean($staffMember['cover_letter'] ));
483                     @$this->   template->   assign('status', ($input)? $this->   template-
>   input('checkbox', 'status', $this->   template->   formatBoolean($staffMember['status'])):$this-
>   template->   formatBoolean($staffMember['status'] ));
```

```
484                 @$this->   template->   assign('employmenttype', ($input)? $this->   template-
>   input('text', 'employmenttype', $staffMember['employmenttype']):$staffMember['employmenttype']);
485                 @$this->   template->   assign('create_date', ($input)? $this->   template-
>   input('text', 'create_date', $staffMember['create_date']):$staffMember['create_date']);
486                 @$this->   template->   assign('create_by', ($input)? $this->   template-
>   input('text', 'create_by', $staffMember['create_by']):$staffMember['create_by']);
487                 @$this->   template->   assign('modify_date', ($input)? $this->   template-
>   input('text', 'modify_date', $staffMember['modify_date']):$staffMember['modify_date']);
488                 @$this->   template->   assign('modify_by', ($input)? $this->   template-
>   input('text', 'modify_by', $staffMember['modify_by']):$staffMember['modify_by']);
489                 @$this->   template->   assign('delete_date', ($input)? $this->   template-
>   input('text', 'delete_date', $staffMember['delete_date']):$staffMember['delete_date']);
490                 @$this->   template->   assign('delete_by', ($input)? $this->   template-
>   input('text', 'delete_by', $staffMember['delete_by']):$staffMember['delete_by']);
491                 @$this->   template->   assign('question_id', ($input)? $this->   template-
>   input('text', 'question_id', $staffMember['question_id']):$staffMember['question_id']);
492                 @$this->   template->   assign('tracking_id', ($input)? $this->   template-
>   input('text', 'tracking_id', $staffMember['tracking_id']):$staffMember['tracking_id']);
493
494
495                 /*if(isset($id)){
496                     $this->template->assign('COMMENTS', $this->comment-
>getCommentBox($id, 'advertisment'));
497                 }*/
498
499       }
500
501       public function Validate($request){
502
503           unset($this->   valid_field);
504           unset($this->   validation_error);
505           $isvalid = True;
506
507           $validfields = $this->   fields;
508           $requiredfields = $this->   fields_required;
509           $fieldsvalidationtype = $this->   fields_validation_type;
510
511           foreach ($request AS $key=>   $value){ //lets strip put unwanted or security violation
fields
512               if(in_array($key, $validfields)){
513                   $this->   valid_field[$key] = $value; //pure fields
514               }
515           }
516
517           foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxs
518               if(!isset($this->   valid_field[$value])){
519                   $this->   valid_field[$value] = '';
520               }
521           }
522
523           if(count($requiredfields) >    0 ){
524               foreach($requiredfields AS $value){ // lets check all the required fields have a value
525                   if (empty($this->   valid_field[$value]) || $this->   valid_field[$value] ==
'NULL'){
526                       $this->   validation_error[$value][] = 'Field is Required'; //error field
527                       $isvalid = false;
528                   }
529               }
530           }
531
532
533
534           //now lets validate
535           foreach ($this->   valid_field AS $key=>   $value){
536               $value = trim($value);
537               if(!empty($value)){ // don't cheak if empty, alread done in required check
538
539                   switch(@$fieldsvalidationtype[$key]){
540                       case 'TEXTAREA': if (strlen($value) >    1024) {
541                                   $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
542                                   $isvalid = false;
543                               } break;
544                       case 'TEXT': if (strlen($value) >    1024) {
545                                   $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
546                                   $isvalid = false;
547                               } break;
548                       case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
```

```
549                                     $this->   validation_error[$key][] = 'not a valid SAP number';
550                                     $isvalid = false;
551                                 } break;
552                         case 'DECIMAL': if (!is_numeric($value)) {
553                                     $this->   validation_error[$key][] = 'Decimal value expected';
554                                     $isvalid = false;
555                                 } break;
556                         case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"          ) && (       $value != 1)) {
557                                     $this->   validation_error[$key][] = 'Please check';
558                                     $isvalid = false;
559                                 } break;
560                         case 'INT': if (!is_numeric($value) &&         $value != 'NULL' ){
561                                     $this->   validation_error[$key][] = 'Numeric value expected';
562                                     $isvalid = false;
563                                 } break;
564                         case 'DATE': //$date = str_replace('/', '-', $value);
565                                      //$date = str_replace("\\", '-', $date);
566                                     @list($day, $month, $year) = explode('/', $value);
567                                     if(!checkdate($month,$day, $year)){
568                                         $this->   validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
569                                         $isvalid = false;
570                                     } break;
571                         case 'YEAR':  if(!checkYear($value)){
572                                         $this->   validation_error[$key][] = 'incorrect year
format, expecting yyyy';
573                                         $isvalid = false;
574                                    } break;
575
576                     }
577
578
579                 }
580             }
581
582             return $isvalid;
583
584         }
585     }
```

File Source for application.class.php

Documentation for this file is available at application.class.php

```
1     <?php
2
3     class application {
4
5         private $db_connect;
6         private $db;
7         public $lastError;
8         public $table;
9         public $template;
10        private $error;
11        public $admin;
12        private $fields =array('application_id', 'applicant_id', 'job_id', 'catagory_id', 'viewed',
'contact_type_id', 'last_contact', 'referral_id', 'status_id', 'contact_susccessful', 'offer_successful',
'reciept_successful', 'saved', 'coverletter_file', 'coverletter_type', 'coverletter_text', 'resume_file',
'resume_type', 'create_date', 'modify_date', 'delete_date');
13        private $fields_required = array('applicant_id', 'job_id', 'catagory_id');
14        private $fields_validation_type = array ('application_id'=>   'INT', 'applicant_id'=>   'INT',
'job_id'=>   'INT', 'catagory_id'=>   'INT', 'viewed'=>   'BOOL', 'contact_type_id'=>   'INT',
'last_contact'=>   'TEXT', 'referral_id'=>   'INT', 'status_id'=>   'INT',
'contact_susccessful'=>   'BOOL', 'offer_successful'=>   'BOOL', 'reciept_successful'=>   'BOOL',
'saved'=>   'BOOL', 'coverletter_file'=>   'TEXT', 'coverletter_type'=>   'TEXT',
'coverletter_text'=>   'TEXT', 'resume_file'=>   'TEXT', 'resume_type'=>   'TEXT',
'create_date'=>   'TEXT', 'modify_date'=>   'TEXT', 'delete_date'=>   'TEXT');
15
16        function __construct(){
17            $this->   db = new db();
18
19            try {
20                $this->   db_connect = $this->   db->   dbh;
21            } catch (CustomException $e) {
22                $e->   logError();
23            }
24
25            $this->   table = new table();
26            $this->   template = new template();
27
28        }
29
30        Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
31
32            $sql = "SELECT application_id,
33    applicant_id,
34    job_id,
35    catagory_id,
36    viewed,
37    contact_type_id,
38    last_contact,
39    referral_id,
40    status_id,
41    contact_susccessful,
42    offer_successful,
43    reciept_successful,
44    saved,
45    coverletter_file,
46    coverletter_type,
47    coverletter_text,
48    resume_file,
49    resume_type,
50    create_date,
51    modify_date,
52    delete_date FROM application WHERE (delete_date ='00-00-0000 00:00:00' OR delete_date IS
NULL)"     ;
53
54            if(is_array($filter)){
55                foreach($filter AS $key=>   $value){
56                    if ($value != 'NULL'  && !empty(        $value)){
57                        $sql .=  " AND "          . $value;
```

```
58                      }
59                  }
60              }
61
62              if($orderby){
63                   $sql .= " ORDER BY "          . $orderby." "          .$direction;
64              }
65
66              try{
67                   $result = $this->   db->   select($sql);
68              }catch(CustomException $e){
69                   echo $e->   queryError($sql);
70              }
71
72              return $result;
73          }
74
75          Private function create($source){
76                  try{
77                      $this->   db_connect->   beginTransaction();
78
79                      foreach($source['application'] AS $key=>   $val){
80                          $field[] = $key;
81                          $value[] = ":"          .$key;
82                      }
83
84                      $sql = "INSERT INTO application ("          .implode(', ',$field).") VALUES
(".implode(', ',$value).");"          ;
85
86                      foreach($source['application'] AS $key=>   $val){
87                          $exec[":"          .$key] = $val;
88                      }
89
90                      try{
91                          $pid = $this->   db->   insert($sql, $exec);
92                      }catch(CustomException $e){
93                          throw new CustomException($e->   queryError($sql));
94                      }
95
96                      $pid = $this->   db_connect->   lastInsertId();
97                  }
98
99                  catch (CustomException $e) {
100                     $e->   queryError($sql);
101                     $this->   db_connect->   rollBack();
102                     return false;
103                 }
104
105                 return $pid;
106         }
107
108
109         Private function read($id){
110
111             $sql = "SELECT application_id,
112     applicant_id,
113     job_id,
114     catagory_id,
115     viewed,
116     contact_type_id,
117     last_contact,
118     referral_id,
119     status_id,
120     contact_susccessful,
121     offer_successful,
122     reciept_successful,
123     saved,
124     coverletter_file,
125     coverletter_type,
126     coverletter_text,
127     resume_file,
128     resume_type,
129     create_date,
130     modify_date,
131     delete_date FROM application WHERE application_id = "      . $id ." AND (delete_date ='00-00-
0000 00:00:00' OR delete_date IS NULL)"       ;
132
133
134                 $stmt = $this->   db_connect->   prepare($sql);
135                 $stmt->   execute();
```

```
136
137                 try{
138                     $result = $this->   db->   select($sql);
139                 }catch(CustomException $e){
140                     echo $e->   queryError($sql);
141                 }
142
143                 return $result[0];
144
145
146         }
147
148         Private function update($source, $id){
149                 try{
150                     $this->   db_connect->   beginTransaction();
151
152                     foreach($source['application'] AS $key=>   $val){
153                         $field[] = $key." = :"          .$key;
154                     }
155
156                     $sql = "UPDATE application SET "          .implode(', ',$field)." WHERE
application_id ="     . $id;
157
158                     foreach($source['application'] AS $key=>   $val){
159                         $exec[":"          .$key] = $val;
160                     }
161
162                     try{
163                             $pid = $this->   db->   update($sql, $exec);
164                     }catch(CustomException $e){
165                             throw new CustomException($e->   queryError($sql));
166                     }
167
168                     $pid = $this->   db_connect->   lastInsertId();
169                 }
170
171                 catch (CustomException $e) {
172                     $e->   queryError($sql);
173                     $this->   db_connect->   rollBack();
174                     return false;
175                 }
176
177                 header('Location:application.php?action=show&id='    .$id);
178
179         }
180
181         Private function remove($id){
182                 if(empty($id)){
183                     return false;
184                 }
185
186                 $sql = "UPDATE application SET delete_date=NOW() WHERE application_id ="          . $id;
187
188                 try{
189                     $result = $this->   db->   update($sql);
190                 }catch(CustomException $e){
191                     echo $e->   queryError($sql);
192                 }
193                 return true;
194         }
195
196
197         /****************** END CRUD METHOD************************/
198
199         public function getApplicationList($type='TABLE',$orderby=NULL, $direction='ASC', $filter=NULL){
200
201             $result = $this->   lists($orderby, $direction, $filter);
202
203             switch(strtoupper($type)){
204
205                 case 'AJAX' : $this->   table->   setRowsOnly();
206                               $this->   table->   removeColumn(array('application_id'));
207                               $this->   table->   setIdentifier('application_id');
208                               $this->   table->   setIdentifierPage('application');
209                               echo $this->   table->   genterateDisplayTable($result);
210
211                     BREAK;
212                 case 'TABLE' :
213                 DEFAULT :
214                     $this->   table->   setHeader(array(
```

```
                        'application_id'=>   'Application Id',
    'applicant_id'=>   'Applicant Id',
    'job_id'=>   'Job Id',
    'catagory_id'=>   'Catagory Id',
    'viewed'=>   'Viewed',
    'contact_type_id'=>   'Contact Type Id',
    'last_contact'=>   'Last Contact',
    'referral_id'=>   'Referral Id',
    'status_id'=>   'Status Id',
    'contact_susccessful'=>   'Contact Susccessful',
    'offer_successful'=>   'Offer Successful',
    'reciept_successful'=>   'Reciept Successful',
    'saved'=>   'Saved',
    'coverletter_file'=>   'Coverletter File',
    'coverletter_type'=>   'Coverletter Type',
    'coverletter_text'=>   'Coverletter Text',
    'resume_file'=>   'Resume File',
    'resume_type'=>   'Resume Type',
    'create_date'=>   'Create Date',
    'modify_date'=>   'Modify Date',
    'delete_date'=>   'Delete Date'));

                    $this->   table->   setFilter(array(
                        'application_id'=>   'TEXT',
    'applicant_id'=>   'TEXT',
    'job_id'=>   'TEXT',
    'catagory_id'=>   'TEXT',
    'viewed'=>   'TEXT',
    'contact_type_id'=>   'TEXT',
    'last_contact'=>   'TEXT',
    'referral_id'=>   'TEXT',
    'status_id'=>   'TEXT',
    'contact_susccessful'=>   'TEXT',
    'offer_successful'=>   'TEXT',
    'reciept_successful'=>   'TEXT',
    'saved'=>   'TEXT',
    'coverletter_file'=>   'TEXT',
    'coverletter_type'=>   'TEXT',
    'coverletter_text'=>   'TEXT',
    'resume_file'=>   'TEXT',
    'resume_type'=>   'TEXT',
    'create_date'=>   'TEXT',
    'modify_date'=>   'TEXT',
    'delete_date'=>   'TEXT'));

                    $this->   table->   removeColumn(array('application_id'));

                    $this->   table->   setIdentifier('application_id');

                    $this->   template->   content($this->   table->   genterateDisplayTable($result));

                    $this->   template->   display();
            }
        }

        Public function showApplicationDetails($id, $return=false){
            $staffMember = $this->   read($id);

            $this->   template->   page('application.tpl.html');

            $this->   templateApplicationLayout($staffMember);

            //if($this->checkAdminLevel(1)){
                $this->   template->   assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='application.php?action=edit&id="                .$id."'\">Edit</div
>"        );
            //}

            echo $this->   template->   fetch();
        }

        Public function editApplicationDetails($id){

            $staffMember = $this->   read($id);

            $name = 'editapplication';

            $this->   template->   page('application.tpl.html');
            $this->   template->   assign('FORM-HEADER', '<form
action="application.php?action=update&id='          .$id.'" method="POST"
```

```
name="'     .$name.'">'        );
292
293           $this->   templateApplicationLayout($staffMember, true);
294
295           $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='application.php?action=show&id=         "     .$id."'\">Cancel</d
iv>"        );
296
297           $this->   template->   display();
298       }
299
300
301       Public function updateApplicationDetails($id){
302
303           if ($this->   Validate($_REQUEST)){
304
305                   $request = $_REQUEST;
306                   $table = 'application';
307
308                   $save[$table]['applicant_id'] = $request['applicant_id'];$save[$table]['job_id'] =
$request['job_id'];$save[$table]['catagory_id'] = $request['catagory_id'];$save[$table]['viewed'] =
$request['viewed'];$save[$table]['contact_type_id'] =
$request['contact_type_id'];$save[$table]['last_contact'] =
$request['last_contact'];$save[$table]['referral_id'] =
$request['referral_id'];$save[$table]['status_id'] =
$request['status_id'];$save[$table]['contact_susccessful'] =
$request['contact_successful'];$save[$table]['offer_successful'] =
$request['offer_successful'];$save[$table]['reciept_successful'] =
$request['reciept_successful'];$save[$table]['saved'] =
$request['saved'];$save[$table]['coverletter_file'] =
$request['coverletter_file'];$save[$table]['coverletter_type'] =
$request['coverletter_type'];$save[$table]['coverletter_text'] =
$request['coverletter_text'];$save[$table]['resume_file'] =
$request['resume_file'];$save[$table]['resume_type'] = $request['resume_type'];
309                   $save[$table]['modify_date'] = date('Y-m-d h:i:s');
310
311                   $this->   update($save, $id );
312
313               }else{
314
315                   $staffMember = $this->   valid_field;
316                   $error = $this->   validation_error;
317
318                   $name = 'editgrant';
319
320                   $this->   template->   page('application.tpl.html');
321
322                   foreach($validfields AS $value){
323                       if(isset($error[$value])){
324                           $this->   template->   assign('err_'.$value, "<span
class=\"error\">"                .implode(',', $error[$value])."</spam>"                );
325                       }
326                   }
327
328                   $this->   template->   assign('FORM-HEADER', '<form
action="application.php?action=update&id='         .$id.'" method="POST"
name="'     .$name.'">'        );
329
330                   $this->   templateApplicationLayout($staffMember, true);
331
332                   if($this->   admin->   checkAdminLevel(1)){
333                       $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.              $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='application.php?action=show&id=         "     .$id."'\">Cancel</d
iv>"        );
334                   }
335                   $this->   template->   assign('FORM-FOOTER', '</form>'      );
336
337                   $this->   template->   display();
338           }
339       }
340
341
342       Public function createApplicationDetails(){
343
344           $name = 'createAdmin';
345
```

```
346             $this->   template->   page('application.tpl.html');
347             $this->   template->   assign('FORM-HEADER', '<form
action="application.php?action=save" method="POST" name="'                          .$name.'">'        );
348
349             $this->   templateApplicationLayout('', true);
350
351             $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Save</div><div
class=\"button\"
onclick=\"location.href='application.php?action=list'\">Cancel</div>                  "     );
352
353
354             $this->   template->   display();
355         }
356
357         Public function saveApplicationDetails(){
358
359             if ($this->   Validate($_REQUEST)){
360
361                     $request = $_REQUEST;
362                     $table = 'application';
363
364                     $save[$table]['applicant_id'] = $request['applicant_id'];$save[$table]['job_id'] =
$request['job_id'];$save[$table]['catagory_id'] = $request['catagory_id'];$save[$table]['viewed'] =
$request['viewed'];$save[$table]['contact_type_id'] =
$request['contact_type_id'];$save[$table]['last_contact'] =
$request['last_contact'];$save[$table]['referral_id'] =
$request['referral_id'];$save[$table]['status_id'] =
$request['status_id'];$save[$table]['contact_susccessful'] =
$request['contact_susccessful'];$save[$table]['offer_successful'] =
$request['offer_successful'];$save[$table]['reciept_successful'] =
$request['reciept_successful'];$save[$table]['saved'] =
$request['saved'];$save[$table]['coverletter_file'] =
$request['coverletter_file'];$save[$table]['coverletter_type'] =
$request['coverletter_type'];$save[$table]['coverletter_text'] =
$request['coverletter_text'];$save[$table]['resume_file'] =
$request['resume_file'];$save[$table]['resume_type'] = $request['resume_type'];
365                     $save[$table]['create_date'] = date('Y-m-d h:i:s');
366
367                     $this->   create($save);
368
369                 }else{
370
371                     $staffMember = $this->   valid_field;
372
373                     $error = $this->   validation_error;
374
375                     $name = 'createapplication';
376
377                     $this->   template->   page('application.tpl.html');
378
379                     foreach($error AS $key=>   $value){
380                         $this->   template->   assign('err_'.$key, "<span
class=\"error\">"                   .@implode(',', $value)."</spam>"                );
381                     }
382
383                     $this->   template->   assign('FORM-HEADER', '<form
action="application.php?action=save" method="POST" name="'                          .$name.'">'        );
384
385                     $this->   templateApplicationLayout($staffMember, true);
386
387                     if($this->   admin->   checkAdminLevel(1)){
388                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.                $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='application.php\">Cancel</div>                  "     );
389                     }
390                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
391
392                     $this->   template->   display();
393             }
394         }
395
396         Public function deleteClientsDetails($id){
397             $this->   remove($id);
398             header('Location: application.php');
399         }
400
401         private function templateApplicationLayout($staffMember, $input = false, $inputArray=array() ){
402
```

```
403                 $id = $staffMember['application_id'];
404
405                                 @$this->   template->   assign('application_id', ($input)? $this-
>   template->   input('text', 'application_id',
$staffMember['application_id']):$staffMember['application_id']);
406                 @$this->   template->   assign('applicant_id', ($input)? $this->   template-
>   input('text', 'applicant_id', $staffMember['applicant_id']):$staffMember['applicant_id']);
407                 @$this->   template->   assign('job_id', ($input)? $this->   template-
>   input('text', 'job_id', $staffMember['job_id']):$staffMember['job_id']);
408                 @$this->   template->   assign('catagory_id', ($input)? $this->   template-
>   input('text', 'catagory_id', $staffMember['catagory_id']):$staffMember['catagory_id']);
409                 @$this->   template->   assign('viewed', ($input)? $this->   template-
>   input('checkbox', 'viewed', $this->   template->   formatBoolean($staffMember['viewed'])):$this-
>   template->   formatBoolean($staffMember['viewed'] ));
410                 @$this->   template->   assign('contact_type_id', ($input)? $this->   template-
>   input('text', 'contact_type_id', $staffMember['contact_type_id']):$staffMember['contact_type_id']);
411                 @$this->   template->   assign('last_contact', ($input)? $this->   template-
>   input('text', 'last_contact', $staffMember['last_contact']):$staffMember['last_contact']);
412                 @$this->   template->   assign('referral_id', ($input)? $this->   template-
>   input('text', 'referral_id', $staffMember['referral_id']):$staffMember['referral_id']);
413                 @$this->   template->   assign('status_id', ($input)? $this->   template-
>   input('text', 'status_id', $staffMember['status_id']):$staffMember['status_id']);
414                 @$this->   template->   assign('contact_susccessful', ($input)? $this->   template-
>   input('checkbox', 'contact_susccessful', $this->   template-
>   formatBoolean($staffMember['contact_susccessful'])):$this->   template-
>   formatBoolean($staffMember['contact_susccessful'] ));
415                 @$this->   template->   assign('offer_successful', ($input)? $this->   template-
>   input('checkbox', 'offer_successful', $this->   template-
>   formatBoolean($staffMember['offer_successful'])):$this->   template-
>   formatBoolean($staffMember['offer_successful'] ));
416                 @$this->   template->   assign('reciept_successful', ($input)? $this->   template-
>   input('checkbox', 'reciept_successful', $this->   template-
>   formatBoolean($staffMember['reciept_successful'])):$this->   template-
>   formatBoolean($staffMember['reciept_successful'] ));
417                 @$this->   template->   assign('saved', ($input)? $this->   template-
>   input('checkbox', 'saved', $this->   template->   formatBoolean($staffMember['saved'])):$this-
>   template->   formatBoolean($staffMember['saved'] ));
418                 @$this->   template->   assign('coverletter_file', ($input)? $this->   template-
>   input('text', 'coverletter_file', $staffMember['coverletter_file']):$staffMember['coverletter_file']);
419                 @$this->   template->   assign('coverletter_type', ($input)? $this->   template-
>   input('text', 'coverletter_type', $staffMember['coverletter_type']):$staffMember['coverletter_type']);
420                 @$this->   template->   assign('coverletter_text', ($input)? $this->   template-
>   input('text', 'coverletter_text', $staffMember['coverletter_text']):$staffMember['coverletter_text']);
421                 @$this->   template->   assign('resume_file', ($input)? $this->   template-
>   input('text', 'resume_file', $staffMember['resume_file']):$staffMember['resume_file']);
422                 @$this->   template->   assign('resume_type', ($input)? $this->   template-
>   input('text', 'resume_type', $staffMember['resume_type']):$staffMember['resume_type']);
423                 @$this->   template->   assign('create_date', ($input)? $this->   template-
>   input('text', 'create_date', $staffMember['create_date']):$staffMember['create_date']);
424                 @$this->   template->   assign('modify_date', ($input)? $this->   template-
>   input('text', 'modify_date', $staffMember['modify_date']):$staffMember['modify_date']);
425                 @$this->   template->   assign('delete_date', ($input)? $this->   template-
>   input('text', 'delete_date', $staffMember['delete_date']):$staffMember['delete_date']);
426
427
428                 /*if(isset($id)){
429                     $this->template->assign('COMMENTS', $this->comment-
>getCommentBox($id,  'application'));
430                 }*/
431
432       }
433
434       public function Validate($request){
435
436           unset($this->   valid_field);
437           unset($this->   validation_error);
438           $isvalid = True;
439
440           $validfields = $this->   fields;
441           $requiredfields = $this->   fields_required;
442           $fieldsvalidationtype = $this->   fields_validation_type;
443
444           foreach ($request AS $key=>   $value){ //lets strip put unwanted or security violation
fields
445               if(in_array($key, $validfields)){
446                   $this->   valid_field[$key] = $value; //pure fields
447               }
448           }
449
450           foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
```

```
though so we can check them, helps with checkboxs
451                 if(!isset($this->   valid_field[$value])){
452                     $this->   valid_field[$value] = '';
453                 }
454             }
455
456             if(count($requiredfields) >    0 ){
457                 foreach($requiredfields AS $value){ // lets check all the required fields have a value
458                     if (empty($this->   valid_field[$value]) || $this->   valid_field[$value] ==
'NULL'){
459                         $this->   validation_error[$value][] = 'Field is Required'; //error field
460                         $isvalid = false;
461                     }
462                 }
463             }
464
465
466
467             //now lets validate
468             foreach ($this->   valid_field AS $key=>   $value){
469                 $value = trim($value);
470                 if(!empty($value)){ // don't cheak if empty, alread done in required check
471
472                     switch(@$fieldsvalidationtype[$key]){
473                         case 'TEXTAREA': if (strlen($value) >    1024) {
474                                             $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
475                                             $isvalid = false;
476                                         } break;
477                         case 'TEXT': if (strlen($value) >    1024) {
478                                             $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
479                                             $isvalid = false;
480                                         } break;
481                         case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
482                                             $this->   validation_error[$key][] = 'not a valid SAP number';
483                                             $isvalid = false;
484                                         } break;
485                         case 'DECIMAL': if (!is_numeric($value)) {
486                                             $this->   validation_error[$key][] = 'Decimal value expected';
487                                             $isvalid = false;
488                                         } break;
489                         case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"          ) && (        $value != 1)) {
490                                             $this->   validation_error[$key][] = 'Please check';
491                                             $isvalid = false;
492                                         } break;
493                         case 'INT': if (!is_numeric($value) &&         $value != 'NULL' ){
494                                             $this->   validation_error[$key][] = 'Numeric value expected';
495                                             $isvalid = false;
496                                         } break;
497                         case 'DATE': $date = str_replace('/', '-', $value);
498                                      $date = str_replace("\\"          , '-', $date);
499                                         @list($day, $month, $year) = explode('-', $date);
500                                         if(!checkdate($month,$day, $year)){
501                                             $this->   validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
502                                             $isvalid = false;
503                                         } break;
504                         case 'YEAR':  if(!checkYear($value)){
505                                             $this->   validation_error[$key][] = 'incorrect year
format, expecting yyyy';
506                                             $isvalid = false;
507                                       } break;
508
509                     }
510                 }
511             }
512
513             return $isvalid;
514
515         }
516     }
```

File Source for template.class.php

Documentation for this file is available at [template.class.php]

```
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,
'r'), filesize(DIR_ROOT."/templates/"          .$layout));
31            }else{
32                $this->   template->   layout = fread(fopen(
DIR_ROOT."/templates/layout.tpl.html"          , 'r'),
filesize(DIR_ROOT."/templates/layout.tpl.html"          ));
33            }
34            //$this->template->layout = str_replace("{*SITE_TYPE*}", SITE_TYPE, $this-
>template->layout );
35        }
36
37        public function __destruct(){
38
39        }
40
41        public function page($field){
42            $val = fread(fopen( DIR_ROOT."/templates/"          .$field, 'r'),
filesize(DIR_ROOT."/templates/"          .$field));
43
44            $this->   template->   layout = str_replace("{*CONTENT*}"          , $val, $this-
>   template->   layout );
45        }
46
47        public function insert($template){
48            $this->   chunk = fread(fopen( DIR_ROOT."/templates/"          .$template, 'r'),
filesize(DIR_ROOT."/templates/"          .$template));
49            return $this->   chunk;
50        }
51
52        public function content($field){
53
54            $this->   template->   layout = str_replace("{*CONTENT*}"          , $field, $this-
>   template->   layout );
55        }
56
57        public function assign($field, $value, &    $tpl=NULL ){
58
59            if($tpl){
```

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

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

File Source for division.class.php

Documentation for this file is available at division.class.php

```
1     <?php
2
3     class division {
4
5         private $db_connect;
6         private $db;
7         public $lastError;
8         public $table;
9         public $template;
10        private $error;
11        public $admin;
12        private $fields =array('division_id', 'name', 'description', 'create_date', 'modify_date',
'delete_date');
13        private $fields_required =array('name');
14        private $fields_validation_type = array ('division_id'=>   'INT', 'name'=>   'TEXT',
'description'=>   'TEXT', 'create_date'=>   'TEXT', 'modify_date'=>   'TEXT', 'delete_date'=>   'TEXT');
15
16        function __construct(){
17            $this->   db = new db();
18
19            try {
20                $this->   db_connect = $this->   db->   dbh;
21            } catch (CustomException $e) {
22                $e->   logError();
23            }
24
25            $this->   table = new table();
26            $this->   template = new template();
27
28        }
29
30        Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
31
32            $sql = "SELECT division_id,
33                        name,
34                        description,
35                        create_date,
36                        modify_date,
37                        delete_date FROM division WHERE (delete_date ='00-00-0000 00:00:00' OR
delete_date IS NULL)"     ;
38
39            if(is_array($filter)){
40                  foreach($filter AS $key=>   $value){
41                      if ($value != 'NULL'  && !empty(        $value)){
42                          $sql .=  " AND "          . $value;
43                      }
44                  }
45            }
46
47            if($orderby){
48                  $sql .= " ORDER BY "          . $orderby." "          .$direction;
49            }
50
51            try{
52                 $result = $this->   db->   select($sql);
53            }catch(CustomException $e){
54                 echo $e->   queryError($sql);
55            }
56
57            return $result;
58        }
59
60        Private function create($source){
61                try{
62                    $this->   db_connect->   beginTransaction();
63
64                    foreach($source['division'] AS $key=>   $val){
```

```
65                      $field[] = $key;
66                      $value[] = ":"          .$key;
67                  }
68
69                  $sql = "INSERT INTO division ("          .implode(', ',$field).") VALUES
(".implode(', ',$value).");"          ;
70
71                  foreach($source['division'] AS $key=>   $val){
72                      $exec[":"          .$key] = $val;
73                  }
74
75                  try{
76                      $pid = $this->   db->   insert($sql, $exec);
77                  }catch(CustomException $e){
78                      throw new CustomException($e->   queryError($sql));
79                  }
80
81                  $pid = $this->   db_connect->   lastInsertId();
82                  $this->   db_connect->   commit();
83              }
84
85              catch (CustomException $e) {
86                  $e->   queryError($sql);
87                  $this->   db_connect->   rollBack();
88                  return false;
89              }
90
91              return $pid;
92      }
93
94
95      Private function read($id){
96
97          $sql = "SELECT division_id,
98      name,
99      description,
100     create_date,
101     modify_date,
102     delete_date FROM division WHERE division_id = "     . $id ." AND (delete_date ='00-00-0000
00:00:00' OR delete_date IS NULL)"      ;
103
104
105             $stmt = $this->   db_connect->   prepare($sql);
106             $stmt->   execute();
107
108             try{
109                 $result = $this->   db->   select($sql);
110             }catch(CustomException $e){
111                 echo $e->   queryError($sql);
112             }
113
114             return $result[0];
115
116
117     }
118
119     Private function update($source, $id){
120             try{
121                 $this->   db_connect->   beginTransaction();
122
123                 foreach($source['division'] AS $key=>   $val){
124                     $field[] = $key." = :"          .$key;
125                 }
126
127                 $sql = "UPDATE division SET "          .implode(', ',$field)." WHERE
division_id ="     . $id;
128
129                 foreach($source['division'] AS $key=>   $val){
130                     $exec[":"          .$key] = $val;
131                 }
132
133                 try{
134                         $pid = $this->   db->   update($sql, $exec);
135                 }catch(CustomException $e){
136                         throw new CustomException($e->   queryError($sql));
137                 }
138
139                 $pid = $this->   db_connect->   lastInsertId();
140             }
141
```

```
142                 catch (CustomException $e) {
143                     $e->   queryError($sql);
144                     $this->   db_connect->   rollBack();
145                     return false;
146                 }
147 
148                 header('Location:division.php?action=show&id='    .$id);
149 
150         }
151 
152         Private function remove($id){
153                 if(empty($id)){
154                     return false;
155                 }
156 
157                 $sql = "UPDATE division SET delete_date=NOW() WHERE division_id ="          . $id;
158 
159                 try{
160                     $result = $this->   db->   update($sql);
161                 }catch(CustomException $e){
162                     echo $e->   queryError($sql);
163                 }
164                 return true;
165         }
166 
167 
168         /****************** END CRUD METHOD************************/
169 
170         public function getDivisionList($type='TABLE',$orderby=NULL, $direction='ASC', $filter=NULL){
171 
172             $result = $this->   lists($orderby, $direction, $filter);
173 
174             switch(strtoupper($type)){
175 
176                 case 'AJAX' : $this->   table->   setRowsOnly();
177                               $this->   table->   removeColumn(array('division_id'));
178                               $this->   table->   setIdentifier('division_id');
179                               $this->   table->   setIdentifierPage('division');
180                               echo $this->   table->   genterateDisplayTable($result);
181 
182                     BREAK;
183                 case 'TABLE' :
184                 DEFAULT :
185                     $this->   table->   setHeader(array(
186                             'division_id'=>   'Division Id',
187                             'name'=>   'Name',
188                             'description'=>   'Description',
189                             'create_date'=>   'Create Date',
190                             'modify_date'=>   'Modify Date',
191                             'delete_date'=>   'Delete Date'));
192 
193                     $this->   table->   setFilter(array(
194                             'division_id'=>   'TEXT',
195                             'name'=>   'TEXT',
196                             'description'=>   'TEXT',
197                             'create_date'=>   'TEXT',
198                             'modify_date'=>   'TEXT',
199                             'delete_date'=>   'TEXT'));
200 
201                     $this->   table->   removeColumn(array('division_id'));
202 
203                     $this->   table->   setIdentifier('division_id');
204 
205                     $this->   template->   content(Box($this->   table-
>   genterateDisplayTable($result),'Division List', 'Shows the current listings for the division,
Division are either groups of people, department or areas that have restricated view on the site'));
206 
207                     $this->   template->   display();
208                 }
209         }
210 
211         Public function showDivisionDetails($id, $return=false){
212             $staffMember = $this->   read($id);
213 
214             $this->   template->   page('division.tpl.html');
215 
216             $this->   templateDivisionLayout($staffMember);
217 
218             //if($this->checkAdminLevel(1)){
219                 $this->   template->   assign('FUNCTION', "<div class=\"button\"
```

```
onclick=\"location.href='division.php?action=edit&id="              .$id."'\">Edit</div&gt
;"     );
220           //}
221
222           echo $this->   template->   fetch();
223       }
224
225       Public function editDivisionDetails($id){
226
227           $staffMember = $this->   read($id);
228
229           $name = 'editdivision';
230
231           $this->   template->   page('division.tpl.html');
232           $this->   template->   assign('FORM-HEADER', '<form
action="division.php?action=update&id='          .$id.'" method="POST"
name="'     .$name.'">'        );
233
234           $this->   templateDivisionLayout($staffMember, true);
235
236           $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='division.php?action=show&id=          "     .$id."'\">Cancel</div&
gt;"     );
237
238           $this->   template->   display();
239       }
240
241
242       Public function updateDivisionDetails($id){
243
244           if ($this->   Validate($_REQUEST)){
245
246                   $request = $_REQUEST;
247                   $table = 'division';
248
249                   $save[$table]['name'] = $request['name'];$save[$table]['description'] =
$request['description'];
250                   $save[$table]['modify_date'] = date('Y-m-d h:i:s');
251
252                   $this->   update($save, $id );
253
254               }else{
255
256                   $staffMember = $this->   valid_field;
257                   $error = $this->   validation_error;
258
259                   $name = 'editgrant';
260
261                   $this->   template->   page('division.tpl.html');
262
263                   foreach($validfields AS $value){
264                       if(isset($error[$value])){
265                           $this->   template->   assign('err_'.$value, "<span
class=\"error\">"                .implode(',', $error[$value])."</spam>"                );
266                       }
267                   }
268
269                   $this->   template->   assign('FORM-HEADER', '<form
action="division.php?action=update&id='          .$id.'" method="POST"
name="'     .$name.'">'        );
270
271                   $this->   templateDivisionLayout($staffMember, true);
272
273                   if($this->   admin->   checkAdminLevel(1)){
274                       $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.              $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='division.php?action=show&id=          "     .$id."'\">Cancel</div&
gt;"     );
275                   }
276                   $this->   template->   assign('FORM-FOOTER', '</form>'      );
277
278                   $this->   template->   display();
279           }
280       }
281
282
283       Public function createDivisionDetails(){
```

```
284
285             $name = 'createAdmin';
286
287             $this->   template->   page('division.tpl.html');
288             $this->   template->   assign('FORM-HEADER', '<form
action="division.php?action=save" method="POST" name="'                              .$name.'">'        );
289
290             $this->   templateDivisionLayout('', true);
291
292             $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Save</div><div
class=\"button\"
onclick=\"location.href='division.php?action=list'\">Cancel</div>                    "     );
293
294
295             $this->   template->   display();
296         }
297
298         Public function saveDivisionDetails(){
299
300             if ($this->   Validate($_REQUEST)){
301
302                     $request = $_REQUEST;
303                     $table = 'division';
304
305                     $save[$table]['name'] = $request['name'];
306                     $save[$table]['description'] = $request['description'];
307                     $save[$table]['create_date'] = date('Y-m-d h:i:s');
308
309                     $id = $this->   create($save);
310                     header('Location: division.php?action=show&id='     .$id);
311                 }else{
312
313                     $staffMember = $this->   valid_field;
314
315                     $error = $this->   validation_error;
316
317                     $name = 'createdivision';
318
319                     $this->   template->   page('division.tpl.html');
320
321                     foreach($error AS $key=>   $value){
322                         $this->   template->   assign('err_'.$key, "<span
class=\"error\">"                    .@implode(',', $value)."</spam>"                );
323                     }
324
325                     $this->   template->   assign('FORM-HEADER', '<form
action="division.php?action=save" method="POST" name="'                              .$name.'">'        );
326
327                     $this->   templateDivisionLayout($staffMember, true);
328
329                     //if($this->admin->checkAdminLevel(1)){
330                         $this->   template->   assign('FUNCTION', "     <div
class=\"button\" onclick=\"document.                $name.submit(); return
false\">Update</div><div class=\"button\"
onclick=\"location.href='division.php\">Cancel</div>                      "     );
331                     //}
332                     $this->   template->   assign('FORM-FOOTER', '</form>'      );
333
334                     $this->   template->   display();
335             }
336         }
337
338         Public function deleteClientsDetails($id){
339             $this->   remove($id);
340             header('Location: division.php');
341         }
342
343         private function templateDivisionLayout($staffMember, $input = false, $inputArray=array() ){
344
345                     $id = @$staffMember['division_id'];
346
347                     @$this->   template->   assign('name', ($input)? $this->   template-
>   input('text', 'name', $staffMember['name']):$staffMember['name']);
348                     @$this->   template->   assign('description', ($input)? $this->   template-
>   input('textarea', 'description', $staffMember['description']):$staffMember['description']);
349                     @$this->   template->   assign('create_date', ($input)? $this->   template-
>   input('text', 'create_date', $staffMember['create_date']):$staffMember['create_date']);
350                     @$this->   template->   assign('modify_date', ($input)? $this->   template-
>   input('text', 'modify_date', $staffMember['modify_date']):$staffMember['modify_date']);
```

```
351                 @$this->   template->   assign('delete_date', ($input)? $this->   template-
>   input('text', 'delete_date', $staffMember['delete_date']):$staffMember['delete_date']);
352
353
354                 /*if(isset($id)){
355                     $this->template->assign('COMMENTS', $this->comment-
>getCommentBox($id, 'division'));
356                 }*/
357
358       }
359
360       public function Validate($request){
361
362           unset($this->   valid_field);
363           unset($this->   validation_error);
364           $isvalid = True;
365
366           $validfields = $this->   fields;
367           $requiredfields = $this->   fields_required;
368           $fieldsvalidationtype = $this->   fields_validation_type;
369
370           foreach ($request AS $key=>   $value){ //lets strip put unwanted or security violation
fields
371               if(in_array($key, $validfields)){
372                   $this->   valid_field[$key] = $value; //pure fields
373               }
374           }
375
376           foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxs
377               if(!isset($this->   valid_field[$value])){
378                   $this->   valid_field[$value] = '';
379               }
380           }
381
382           if(count($requiredfields) >    0 ){
383               foreach($requiredfields AS $value){ // lets check all the required fields have a value
384                   if (empty($this->   valid_field[$value]) || $this->   valid_field[$value] ==
'NULL'){
385                       $this->   validation_error[$value][] = 'Field is Required'; //error field
386                       $isvalid = false;
387                   }
388               }
389           }
390
391
392
393           //now lets validate
394           foreach ($this->   valid_field AS $key=>   $value){
395               $value = trim($value);
396               if(!empty($value)){ // don't cheak if empty, alread done in required check
397
398                   switch(@$fieldsvalidationtype[$key]){
399                       case 'TEXTAREA': if (strlen($value) >    1024) {
400                                   $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
401                                   $isvalid = false;
402                               } break;
403                       case 'TEXT': if (strlen($value) >    1024) {
404                                   $this->   validation_error[$key][] = 'Field longer then 1024
charactors';
405                                   $isvalid = false;
406                               } break;
407                       case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
408                                   $this->   validation_error[$key][] = 'not a valid SAP number';
409                                   $isvalid = false;
410                               } break;
411                       case 'DECIMAL': if (!is_numeric($value)) {
412                                   $this->   validation_error[$key][] = 'Decimal value expected';
413                                   $isvalid = false;
414                               } break;
415                       case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"          ) && (        $value != 1)) {
416                                   $this->   validation_error[$key][] = 'Please check';
417                                   $isvalid = false;
418                               } break;
419                       case 'INT': if (!is_numeric($value) &&         $value != 'NULL' ){
420                                   $this->   validation_error[$key][] = 'Numeric value expected';
421                                   $isvalid = false;
422                               } break;
```

```
423                     case 'DATE': $date = str_replace('/', '-', $value);
424                                  $date = str_replace("\\"          , '-', $date);
425                                     @list($day, $month, $year) = explode('-', $date);
426                                     if(!checkdate($month,$day, $year)){
427                                         $this->   validation_error[$key][] = 'incorrect date
format, expecting dd/mm/yyyy';
428                                         $isvalid = false;
429                                     } break;
430                     case 'YEAR':  if(!checkYear($value)){
431                                         $this->   validation_error[$key][] = 'incorrect year
format, expecting yyyy';
432                                         $isvalid = false;
433                                    } break;
434
435                 }
436             }
437         }
438
439         return $isvalid;
440
441     }
442
443
444     public function getSelectListOfDivision($id, $selectBox=NULL){
445
446             if($selectBox == false){
447                 return $this->   template->   getListTable('division', $id, 'division_id', 'name');
448             }else{
449                 $select = "<select name=\"division_id\">"                          ;
450                 $select .= "<option value=\"NULL\"></option>"                              ;
451                 $select .= $this->   template->   getListTable('division', $id, 'division_id',
'name', $selectBox);
452                 $select .= "</select>"               ;
453                 return $select;
454             }
455     }
456 }
```

File Source for login.class.php

Documentation for this file is available at login.class.php

```php
<?php

class login {

    public $lastError;
    public $table;
    public $template;
    private $error;
    public $admin;

    function __construct(){
        $this->   table = new table();
        $this->   template = new template();

    }

    public function getHomePage(){
        $this->   template = new template();
        $this->   template->   page('index.tpl.html');
        echo $this->   template->   display();

    }

    function checkUserLogin($uname, $password){
        try {
            $conn2 = new PDO(MAIN_DB_TYPE.':dbname='.MAIN_DB_DBASE.';host='.MAIN_DB_HOST,
MAIN_DB_USER, MAIN_DB_PASS);
        } catch (PDOException $e) {
            echo $e->   getMessage();
        }

        $conn2->   setAttribute (PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        $sql = "SELECT users.user_id,clients_users.client_id,server_connection.* FROM
                users
                Left Join clients_users ON users.user_id = clients_users.user_id
                Left Join clients_server_connection ON clients_users.client_id =
clients_server_connection.client_id
                Left Join server_connection ON clients_server_connection.connection_id =
server_connection.connection_id
                WHERE user_name='"     .$uname."' AND
user_password='"     .$password."'"          ;

        try{
            $stmt2 = $conn2->   prepare($sql);
        }catch(CustomException $e){
            throw new CustomException('QUERY : '. $e->   getMessage());
        }

        $stmt2->   execute();

        if ($stmt2->   errorCode() != 00000 )
        {
              $error = $stmt2->   errorInfo();
              throw new CustomException($error[2]);
        }

        $result = $stmt2->   fetch(PDO::FETCH_NAMED);

        $_SESSION['dbaccess']['server_host'] = $result['server_host'];
        $_SESSION['dbaccess']['server_port'] = $result['server_port'];
        $_SESSION['dbaccess']['server_database'] = $result['server_database'];
        $_SESSION['dbaccess']['server_type'] = $result['server_type'];
        $_SESSION['user']['client_id'] = $result['client_id'];
        $_SESSION['user']['user_id'] = $result['user_id'];

        pp($_SESSION);
```

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

File Source for question.class.php

Documentation for this file is available at question.class.php

```
1    <?  php
```

File Source for users.class.php

Documentation for this file is available at users.class.php

```
1     <?php
2
3     require_once('administration.class.php');
4     require_once('division.class.php');
5
6     class users {
7
8         private $db_connect;
9         private $db;
10        public $lastError;
11        public $table;
12        public $template;
13        public $administration;
14        public $division;
15        private $error;
16        public $admin;
17        private $fields =array('user_id', 'username', 'password', 'name', 'surname', 'email', 'active',
'last_login', 'division_id', 'administration_id', 'create_date', 'modified_date', 'delete_date');
18        private $fields_required = array('username', 'password', 'name', 'email');
19        private $fields_validation_type = array ('user_id'=>   'INT', 'username'=>   'TEXT',
'password'=>   'TEXT', 'name'=>   'TEXT', 'surname'=>   'TEXT', 'email'=>   'TEXT', 'active'=>   'BOOL',
'last_login'=>   'TEXT', 'division_id'=>   'INT', 'administration_id'=>   'INT',
'create_date'=>   'TEXT', 'modified_date'=>   'TEXT', 'delete_date'=>   'TEXT');
20
21        function __construct(){
22            $this->   db = new db();
23
24            try {
25                $this->   db_connect = $this->   db->   dbh;
26            } catch (CustomException $e) {
27                $e->   logError();
28            }
29
30            $this->   table = new table();
31            $this->   template = new template();
32            $this->   administration = new administration();
33            $this->   division = new division();
34
35        }
36
37        Private function lists($orderby=NULL, $direction='ASC', $filter=NULL){
38
39            $sql = "SELECT     user_id,
40                            username,
41                            password,
42                            users.name,
43                            surname,
44                            email,
45                            active,
46                            DATE_FORMAT(last_login, '%d/%m/%Y') AS last_login,
47                            division.name AS division_name,
48                            group_name,
49                            DATE_FORMAT(users.create_date, '%d/%m/%Y') AS create_date,
50                            DATE_FORMAT(users.modify_date, '%d/%m/%Y') AS modify_date,
51                            DATE_FORMAT(users.delete_date, '%d/%m/%Y') AS delete_date
52                    FROM users
53                        LEFT JOIN administration ON
users.administration_id=administration.administration_id
54                        LEFT JOIN division ON users.division_id=division.division_id
55                    WHERE (users.delete_date ='00-00-0000 00:00:00' OR users.delete_date IS NULL)"     ;
56
57            if(is_array($filter)){
58                foreach($filter AS $key=>   $value){
59                    if ($value != 'NULL'  && !empty(        $value)){
60                        $sql .=  " AND "          . $value;
61                    }
62                }
```

```
63            }
64
65            if($orderby){
66                $sql .= " ORDER BY "          . $orderby." "          .$direction;
67            }
68
69            try{
70                $result = $this->   db->   select($sql);
71            }catch(CustomException $e){
72                echo $e->   queryError($sql);
73            }
74
75            return $result;
76        }
77
78        Private function create($source){
79
80            try {
81                $conn2 = new PDO('mysql:dbname=people_scope_main;host=DEV;port=3306', 'root',
'password');
82            } catch (PDOException $e) {
83                echo $e->   getMessage();
84            }
85
86            $conn2->   setAttribute (PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
87
88            $sql = "INSERT INTO users (user_name, user_password, create_date) VALUES
('"     .$source['users']['username']."','"          .$source['users']['password']."',NOW());"          ;
89
90            try{
91                $stmt2 = $conn2->   prepare($sql);
92            }catch(CustomException $e){
93                throw new CustomException('QUERY : '. $e->   getMessage());
94            }
95
96            $stmt2->   execute();
97
98            if ($stmt2->   errorCode() != 00000 )
99            {
100                $error = $stmt2->   errorInfo();
101                throw new CustomException($error[2]);
102           }
103
104           $source['users']['user_id'] = $conn2->   lastInsertId();
105
106           $sql = "INSERT INTO clients_users (client_id, user_id) VALUES
("     .$_SESSION['user']['client_id'].", "          .$source['users']['user_id'].")"          ;
107           echo $sql;
108           try{
109               $stmt2 = $conn2->   prepare($sql);
110           }catch(CustomException $e){
111               throw new CustomException('QUERY : '. $e->   getMessage());
112           }
113
114           $stmt2->   execute();
115
116           if ($stmt2->   errorCode() != 00000 )
117           {
118                $error = $stmt2->   errorInfo();
119                throw new CustomException($error[2]);
120           }
121
122           try{
123               $this->   db_connect->   beginTransaction();
124
125               foreach($source['users'] AS $key=>   $val){
126                   $field[] = $key;
127                   $value[] = ":"          .$key;
128               }
129
130               $sql = "INSERT INTO users ("          .implode(', ',$field).") VALUES
("     .implode(', ',$value).");"          ;
131
132               foreach($source['users'] AS $key=>   $val){
133                   $exec[":"          .$key] = $val;
134               }
135
136               try{
137                   $pid = $this->   db->   insert($sql, $exec);
138                   $this->   db_connect->   commit();
```

```
139             }catch(CustomException $e){
140                 throw new CustomException($e->   queryError($sql));
141             }
142 
143             $pid = $this->   db_connect->   lastInsertId();
144         }
145 
146         catch (CustomException $e) {
147             $e->   queryError($sql);
148             $this->   db_connect->   rollBack();
149             return false;
150         }
151 
152 
153 
154 
155 
156         return $pid;
157     }
158 
159 
160     Private function read($id){
161 
162         $sql = "SELECT user_id,
163                             username,
164                             password,
165                             users.name,
166                             surname,
167                             email,
168                             active,
169                             last_login,
170                             users.division_id,
171                             division.name AS devision_name,
172                             group_name,
173                             users.administration_id,
174                             users.create_date,
175                             users.modify_date,
176                             users.delete_date FROM users
177                 LEFT JOIN administration ON users.administration_id=administration.administration_id
178                 LEFT JOIN division ON users.division_id=division.division_id
179                 WHERE user_id = "     . $id ." AND (users.delete_date ='00-00-0000 00:00:00'
OR users.delete_date IS NULL)"     ;
180 
181 
182         $stmt = $this->   db_connect->   prepare($sql);
183         $stmt->   execute();
184 
185         try{
186             $result = $this->   db->   select($sql);
187         }catch(CustomException $e){
188             echo $e->   queryError($sql);
189         }
190 
191         return $result[0];
192 
193 
194     }
195 
196     Private function update($source, $id){
197         try{
198             $this->   db_connect->   beginTransaction();
199 
200             foreach($source['users'] AS $key=>   $val){
201                 $field[] = $key." = :"          .$key;
202             }
203 
204             $sql = "UPDATE users SET "          .implode(', ',$field)." WHERE user_id ="          .
$id;
205 
206             foreach($source['users'] AS $key=>   $val){
207                 $exec[":"          .$key] = $val;
208             }
209 
210             try{
211                 $pid = $this->   db->   update($sql, $exec);
212                 $this->   db_connect->   commit();
213             }catch(CustomException $e){
214                 throw new CustomException($e->   queryError($sql));
215             }
216 
```

```
217                 $pid = $this->   db_connect->   lastInsertId();
218
219             }
220
221             catch (CustomException $e) {
222                 $e->   queryError($sql);
223                 $this->   db_connect->   rollBack();
224                 return false;
225             }
226
227             header('Location:users.php?action=show&id='    .$id);
228
229         }
230
231         Private function remove($id){
232             if(empty($id)){
233                 return false;
234             }
235
236             $sql = "UPDATE users SET delete_date=NOW() WHERE user_id ="          . $id;
237
238             try{
239                 $result = $this->   db->   update($sql);
240             }catch(CustomException $e){
241                 echo $e->   queryError($sql);
242             }
243             return true;
244         }
245
246
247         /****************** END CRUD METHOD************************/
248
249         public function getUsersList($type='TABLE',$orderby=NULL, $direction='ASC', $filter=NULL){
250
251             $result = $this->   lists($orderby, $direction, $filter);
252
253             $this->   table->   removeColumn(array('user_id', 'modify_date', 'delete_date'));
254
255             switch(strtoupper($type)){
256
257                 case 'AJAX' : $this->   table->   setRowsOnly();
258                               $this->   table->   setIdentifier('user_id');
259                               $this->   table->   setIdentifierPage('users');
260                               echo $this->   table->   genterateDisplayTable($result);
261
262                 BREAK;
263                 case 'TABLE' :
264                 DEFAULT :
265                     $this->   table->   setHeader(array(
266                             'user_id'=>   'User Id',
267                             'username'=>   'Username',
268                             'password'=>   'Password',
269                             'name'=>   'Name',
270                             'surname'=>   'Surname',
271                             'email'=>   'Email',
272                             'active'=>   'Active',
273                             'last_login'=>   'Login',
274                             'division_name'=>   'Division',
275                             'group_name'=>   'Group',
276                             'create_date'=>   'Created'));
277
278                     $this->   table->   setFilter(array(
279                             'user_id'=>   'TEXT',
280                             'username'=>   'TEXT',
281                             'password'=>   'TEXT',
282                             'name'=>   'TEXT',
283                             'surname'=>   'TEXT',
284                             'email'=>   'TEXT',
285                             'active'=>   'TEXT',
286                             'last_login'=>   'TEXT',
287                             'division_name'=>   'TEXT',
288                             'group_name'=>   'COMPILED',
289                             'create_date'=>   'TEXT'));
290
291                     $this->   table->   setIdentifier('user_id');
292
293                     $this->   template->   content(box($this->   table-
>   genterateDisplayTable($result), "User List"          , "list of all current users"          ));
294
295                     $this->   template->   display();
```

```
296         }
297     }
298
299     Public function showUsersDetails($id, $return=false){
300         $staffMember = $this->   read($id);
301
302         $this->   template->   page('users.tpl.html');
303
304         $this->   templateUsersLayout($staffMember);
305
306         //if($this->checkAdminLevel(1)){
307         $this->   template->   assign('FUNCTION', "<div class=\"button\"
onclick=\"location.href='users.php?action=edit&id="             .$id."'\">Edit</div>&q
uot;);
308         //}
309
310         echo $this->   template->   fetch();
311     }
312
313     Public function editUsersDetails($id){
314
315         $staffMember = $this->   read($id);
316
317         $name = 'editusers';
318
319         $this->   template->   page('users.tpl.html');
320         $this->   template->   assign('FORM-HEADER', '<form
action="users.php?action=update&id='         .$id.'" method="POST"
name="'     .$name.'">'        );
321
322         $this->   templateUsersLayout($staffMember, true);
323
324         $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='users.php?action=show&id=         "     .$id."'\">Cancel</div>
"     );
325
326         $this->   template->   display();
327     }
328
329
330     Public function updateUsersDetails($id){
331
332         if ($this->   Validate($_REQUEST)){
333
334             $request = $_REQUEST;
335             $table = 'users';
336
337             $save[$table]['username'] = $request['username'];
338             $save[$table]['password'] = $request['password'];
339             $save[$table]['name'] = $request['name'];
340             $save[$table]['surname'] = $request['surname'];
341             $save[$table]['email'] = $request['email'];
342             $save[$table]['active'] = $request['active'];
343             $save[$table]['last_login'] = $request['last_login'];
344             $save[$table]['division_id'] = $request['division_id'];
345             $save[$table]['administration_id'] = $request['administration_id'];
346             $save[$table]['modify_date'] = date('Y-m-d h:i:s');
347
348             $this->   update($save, $id );
349
350         }else{
351
352             $staffMember = $this->   valid_field;
353             $error = $this->   validation_error;
354
355             $name = 'editgrant';
356
357             $this->   template->   page('users.tpl.html');
358
359             foreach($validfields AS $value){
360                 if(isset($error[$value])){
361                     $this->   template->   assign('err_'.$value, "<span
class=\"error\">"                 .implode(',', $error[$value])."</spam>"                );
362                 }
363             }
364
365             $this->   template->   assign('FORM-HEADER', '<form
action="users.php?action=update&id='         .$id.'" method="POST"
```

```
name="'     .$name.'">'        );
366
367               $this->   templateUsersLayout($staffMember, true);
368
369               if($this->   admin->   checkAdminLevel(1)){
370                   $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Update</div><div
class=\"button\"
onclick=\"location.href='users.php?action=show&id=         "     .$id."'\">Cancel</div>
"     );
371               }
372               $this->   template->   assign('FORM-FOOTER', '</form>'      );
373
374               $this->   template->   display();
375           }
376       }
377
378
379       Public function createUsersDetails(){
380
381           $name = 'createAdmin';
382
383           $this->   template->   page('users.tpl.html');
384           $this->   template->   assign('FORM-HEADER', '<form
action="users.php?action=save" method="POST" name="'                          .$name.'">'        );
385
386           $this->   templateUsersLayout('', true);
387
388           $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Save</div><div
class=\"button\"
onclick=\"location.href='users.php?action=list'\">Cancel</div>                    "     );
389
390
391           $this->   template->   display();
392       }
393
394       Public function saveUsersDetails(){
395
396           if ($this->   Validate($_REQUEST)){
397
398               $request = $_REQUEST;
399               $table = 'users';
400
401               $save[$table]['active'] = (isset($request['active']))?1:0;
402
403               $save[$table]['username'] = $request['username'];
404               $save[$table]['password'] = $request['password'];
405               $save[$table]['name'] = $request['name'];
406               $save[$table]['surname'] = $request['surname'];
407               $save[$table]['email'] = $request['email'];
408               $save[$table]['active'] = (isset($request['active']))?1:0;
409               $save[$table]['last_login'] = $request['last_login'];
410               $save[$table]['division_id'] = $request['division_id'];
411               $save[$table]['administration_id'] = $request['administration_id'];
412               $save[$table]['create_date'] = date('Y-m-d h:i:s');
413
414               $id = $this->   create($save);
415               header('Location: users.php?action=show&id='     .$id);
416
417           }else{
418
419               $staffMember = $this->   valid_field;
420
421               $error = $this->   validation_error;
422
423               $name = 'createusers';
424
425               $this->   template->   page('users.tpl.html');
426
427               foreach($error AS $key=>   $value){
428                   $this->   template->   assign('err_'.$key, "<span
class=\"error\">"                    .@implode(',', $value)."</spam>"                );
429               }
430
431               $this->   template->   assign('FORM-HEADER', '<form
action="users.php?action=save" method="POST" name="'                          .$name.'">'        );
432
433               $this->   templateUsersLayout($staffMember, true);
434
```

```
435             //if($this->admin->checkAdminLevel(1)){
436             $this->   template->   assign('FUNCTION', "     <div class=\"button\"
onclick=\"document.     $name.submit(); return false\">Save</div><div
class=\"button\" onclick=\"location.href='users.php'\">Cancel</div>                                "     );
437             //}
438             $this->   template->   assign('FORM-FOOTER', '</form>'      );
439
440             $this->   template->   display();
441         }
442     }
443
444     Public function deleteClientsDetails($id){
445         $this->   remove($id);
446         header('Location: users.php');
447     }
448
449     private function templateUsersLayout($staffMember, $input = false, $inputArray=array() ){
450
451         $id = @$staffMember['user_id'];
452
453         /*@$this->template->assign('user_id', ($input)? $this->template->input('text',
'user_id', $staffMember['user_id']):$staffMember['user_id']);*/
454         @$this->   template->   assign('username', ($input)? $this->   template->   input('text',
'username', $staffMember['username']):$staffMember['username']);
455         @$this->   template->   assign('password', ($input)? $this->   template->   input('text',
'password', $staffMember['password']):$staffMember['password']);
456         @$this->   template->   assign('name', ($input)? $this->   template->   input('text',
'name', $staffMember['name']):$staffMember['name']);
457         @$this->   template->   assign('surname', ($input)? $this->   template->   input('text',
'surname', $staffMember['surname']):$staffMember['surname']);
458         @$this->   template->   assign('email', ($input)? $this->   template->   input('text',
'email', $staffMember['email']):$staffMember['email']);
459         @$this->   template->   assign('active', ($input)? $this->   template->   input('checkbox',
'active', $this->   template->   formatBoolean($staffMember['active'])):$this->   template-
>   formatBoolean($staffMember['active'] ));
460         @$this->   template->   assign('last_login', ($input)? $this->   template->   input('text',
'last_login', $staffMember['last_login']):$staffMember['last_login']);
461         $this->   template->   assign('division_id', ($input)?  $this->   division-
>   getSelectListOfDivision($staffMember['division_id'], TRUE):$this->   division-
>   getSelectListOfDivision($staffMember['division_id']));
462         $this->   template->   assign('group_name', ($input)? $this->   administration-
>   getSelectListOfAdministration($staffMember['administration_id'], TRUE):$this->   administration-
>   getSelectListOfAdministration($staffMember['administration_id']));
463
464         /*if(isset($id)){
465          $this->template->assign('COMMENTS', $this->comment->getCommentBox($id,
'users'));
466          }*/
467
468     }
469
470     public function Validate($request){
471
472         unset($this->   valid_field);
473         unset($this->   validation_error);
474         $isvalid = True;
475
476         $validfields = $this->   fields;
477         $requiredfields = $this->   fields_required;
478         $fieldsvalidationtype = $this->   fields_validation_type;
479
480         foreach ($request AS $key=>   $value){ //lets strip put unwanted or security violation
fields
481             if(in_array($key, $validfields)){
482                 $this->   valid_field[$key] = $value; //pure fields
483             }
484         }
485
486         foreach ($validfields AS $value){ //now lets just add fields as blank if they didn't come
though so we can check them, helps with checkboxs
487             if(!isset($this->   valid_field[$value])){
488                 $this->   valid_field[$value] = '';
489             }
490         }
491
492         if(count($requiredfields) >    0 ){
493             foreach($requiredfields AS $value){ // lets check all the required fields have a value
494                 if (empty($this->   valid_field[$value]) || $this->   valid_field[$value] ==
'NULL'){
495                     $this->   validation_error[$value][] = 'Field is Required'; //error field
```

```
496                     $isvalid = false;
497                 }
498             }
499         }
500
501
502
503         //now lets validate
504         foreach ($this->   valid_field AS $key=>   $value){
505             $value = trim($value);
506             if(!empty($value)){ // don't cheak if empty, alread done in required check
507
508                 switch(@$fieldsvalidationtype[$key]){
509                     case 'TEXTAREA': if (strlen($value) >    1024) {
510                         $this->   validation_error[$key][] = 'Field longer then 1024 charactors';
511                         $isvalid = false;
512                     } break;
513                     case 'TEXT': if (strlen($value) >    1024) {
514                         $this->   validation_error[$key][] = 'Field longer then 1024 charactors';
515                         $isvalid = false;
516                     } break;
517                     case 'SAP': if ((!is_numeric($value)) || (strlen($value) != 10)) {
518                         $this->   validation_error[$key][] = 'not a valid SAP number';
519                         $isvalid = false;
520                     } break;
521                     case 'DECIMAL': if (!is_numeric($value)) {
522                         $this->   validation_error[$key][] = 'Decimal value expected';
523                         $isvalid = false;
524                     } break;
525                     case 'BOOL': if ((!is_bool($value)) &&
(strtoupper($value)!="YES"          ) && (        $value != 1)) {
526                         $this->   validation_error[$key][] = 'Please check';
527                         $isvalid = false;
528                     } break;
529                     case 'INT': if (!is_numeric($value) &&         $value != 'NULL' ){
530                         $this->   validation_error[$key][] = 'Numeric value expected';
531                         $isvalid = false;
532                     } break;
533                     case 'DATE': $date = str_replace('/', '-', $value);
534                     $date = str_replace("\\"          , '-', $date);
535                     @list($day, $month, $year) = explode('-', $date);
536                     if(!checkdate($month,$day, $year)){
537                         $this->   validation_error[$key][] = 'incorrect date format, expecting
dd/mm/yyyy';
538                         $isvalid = false;
539                     } break;
540                     case 'YEAR':  if(!checkYear($value)){
541                         $this->   validation_error[$key][] = 'incorrect year format, expecting
yyyy';
542                         $isvalid = false;
543                     } break;
544
545                 }
546             }
547         }
548
549         return $isvalid;
550
551     }
552  }
```

Appendix D - Todo List

In Package PeopleScope

In **form::$autoRefresh**

- should be removed for JQuery ready({})

In **form::$autoRefreshcheckboxs**

- should be removed for JQuery ready({})

In **form::$autoRefreshInputEnter**

- should be removed for JQuery ready({})

In **template::$repeatRegion**

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

In **convertUIdate()**

- remember how this works :S

In **form::formScript()**

- should be removed for JQuery ready({})

In **form::setAutoRefresh()**

- should be removed for JQuery ready({})

In **email::setTXTtemplate()**

- confirm this is correct

In **department::templateDepartmentLayout()**

- find out what $inputArray is used for

In **department2::templateDepartmentLayout()**

- find out what $inputArray is used for

In **form::unsetAutoRefresh()**

- should be removed for JQuery ready({})

Index

A

B

C

D

E

F

*
*

G

L

M

P

Q

S

T

U

W