



API and Functions list

Function Name	Small Description
<u>add_action</u>	Add an action/operation to the list table.
<u>add_fields</u>	The fields that user will see on add operation
<u>callback_add_field</u>	This callback escapes the default auto field output of the field name at the add form.
<u>callback_after_delete</u>	The callback runs when the operation delete completed successfully
<u>callback_after_insert</u>	This is a callback after the auto insert of the CRUD.
<u>callback_after_update</u>	This is a callback that is used after the automatic update of the CRUD.
<u>callback_after_upload</u>	A callback that triggered after the upload functionality.
<u>callback_before_delete</u>	This callback runs before the auto delete of the crud.
<u>callback_before_insert</u>	This callback runs before the auto insert of the crud.
<u>callback_before_update</u>	This callback runs before the auto update of the crud.
<u>callback_before_upload</u>	A callback that triggered before the upload functionality. This callback is suggested for validation checks.
<u>callback_column</u>	This callback runs on each row. It escapes the auto column value and runs the callback.
<u>callback_delete</u>	This callback escapes the auto delete of the CRUD , and runs only the callback.
<u>callback_edit_field</u>	This callback escapes the default auto field output of the field name at the edit form.
<u>callback_field</u>	This callback escapes the default auto field output of the field name for the add and edit form.
<u>callback_insert</u>	This callback escapes the auto insert of the CRUD, and runs only the inserted callback.
<u>callback_update</u>	This callback escapes the auto update of the CRUD , and runs only the callback.
<u>callback_upload</u>	A callback that replaces the default auto uploader.
<u>change_field_type</u>	Changes the default field type

<u>columns</u>	The displayed columns that user see
<u>display_as</u>	Changes the displaying label of the table field
<u>edit_fields</u>	The fields that user will see on edit operation
<u>fields</u>	The fields that user will see on add/edit
<u>field_type</u>	Just an alias to the change_field_type method.
<u>getState</u>	Get the state string key according to the documentation.
<u>getStateInfo</u>	Get the state information for the operation that fired.
<u>get_field_types</u>	Note: Getters is only to view some info and always works after the function render
<u>get_primary_key</u>	Note: Getters is only to view some info and always works after the function render
<u>like</u>	Same as codeigniter's like for the list.
<u>limit</u>	Same as limit of codeigniter for the table list
<u>order_by</u>	A quick first order_by (same as codeigniter) to our list
<u>or_like</u>	Same as or_like of codeigniter for the table list
<u>or_where</u>	Same as codeigniter or_where for the list.
<u>render</u>	Or else ... make it work!! The web application takes decision of what to do and show it to the user.
<u>required_fields</u>	Sets the required fields of add and edit fields.
<u>set_crud_url_path</u>	This method is useful when the path is not specified correctly. Especially when we are using routes.
<u>set_field_upload</u>	Sets a field name to be an uploaded file.
<u>set_language</u>	Simply set the language.
<u>set_lang_string</u>	Set a language string directly.
<u>set_model</u>	Sets the model that crud will use (The model always must extends grocery_Model)
<u>set_primary_key</u>	Handles the default primary key for a specific table.
<u>set_relation</u>	Set a relation 1-n database relation.
<u>set_relation_n_n</u>	Sets a relation with n-n relationship.
<u>set_rules</u>	Set Validation Rules (Same as Codeigniter set_rules)
<u>set_subject</u>	Set a subject to understand what type of CRUD you use.
<u>set_table</u>	Sets the basic database table that we will get our data.
<u>set_theme</u>	Set the CRUD theme - For now on there is only 'flexigrid' and 'datatables' . The default theme is flexigrid.
<u>unset_add</u>	Unsets the add operation

<u>unset_add_fields</u>	unsets the fields at the add form.
<u>unset_back_to_list</u>	Unset all the "back to list" buttons and messages.
<u>unset_columns</u>	Unset columns from the list
<u>unset_delete</u>	Remove the delete operation from the CRUD grid
<u>unset_edit</u>	Unsets the edit operation
<u>unset_edit_fields</u>	unsets the fields at the edit form.
<u>unset_export</u>	Unset the export button and don't let the user to use this functionality.
<u>unset_fields</u>	Unset fields from both add and edit form.
<u>unset_jquery</u>	Unset the JQuery from loading.
<u>unset_jquery_ui</u>	Unset the JQueryUI from loading.
<u>unset_list</u>	Unset the first page list (datagrid)
<u>unset_operations</u>	Unset all the operations . A user have access only to the grid.
<u>unset_print</u>	Unset the print button and don't let the user to use this functionality.
<u>unset_read</u>	Unsets the read operation
<u>unset_texteditor</u>	Unsets the texteditor of the selected fields
<u>where</u>	A quick database where (Same as codeigniter) to our list

add_action

```
void add_action( string $label, string $image_url , string $link_url , string $css_class , mixed $url_callback)
```

Add an action/operation to the list table. The way to do that its simple:

1. add the label of your subject, for example "Photo Gallery".
2. add an image url . Notice that in the flexigrid theme is a required field.
3. add your custom url. This url will be connected with the site_url function and the primary key. For example if you add 'my_controller/photo/' then it will transform to site_url('my_controller/photo/.\$primary_key_for_each_row)
4. add a CSS class. Remember that at the theme of datatables a CSS class includes images , for example : ui-icon-image, ui-icon-plus, e.t.c.
5. You can even add your own url callback. If you don't want the default transformation of the url, you can use this parameter to add your own algorithm of the url transformation.

Example:

```
function offices_management_with_actions()
{
    $crud = new grocery_CRUD();

    $crud->set_theme('datatables');
    $crud->set_table('offices');
    $crud->set_subject('Office');
    $crud->required_fields('city');
    $crud->columns('city','country','phone');

    $crud->add_action('More', '', 'demo/action_more','ui-icon-plus');
    $crud->add_action('Photos', '', '', 'ui-icon-
image',array($this,'just_a_test'));
    $crud->add_action('Smileys',
'http://www.grocerycrud.com/assets/uploads/general/smiley.png',
'demo/action_smiley');

    $output = $crud->render();

    $this->_example_output($output);
}

function just_a_test($primary_key , $row)
{
    return site_url('demo/action/action_photos').'?country='.$row->country;
}
```


add_fields

```
void add_fields( string $var [, string $var [, string $... ]] )
```

The fields that user will see on add operation.

Example:

```
function customers_example() {
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        ->columns('customerName','contactLastName','creditLimit');

    $crud->add_fields('customerName','contactLastName','city','creditLimit');
    $crud->edit_fields('customerName','contactLastName','city');

    $crud->required_fields('customerName','contactLastName');

    $output = $crud->render();

    $this->_example_output($output);
}
```

callback_add_field

```
void callback_add_field( string $field , mixed $callback )
```

This callback escapes the default auto field output of the field name at the add form. There are no parameters to this callback. You must return to the callback function a string.

Example:

```
function example_callback_add_field(){
    $crud = new grocery_CRUD();

    $crud->set_table('offices');
    $crud->set_subject('Office');
    $crud->required_fields('city');
    $crud->columns('city','country','phone','addressLine1','postalCode');

    $crud->callback_add_field('phone',array($this,'add_field_callback_1'));
    $crud->callback_add_field('state',array($this,'add_field_callback_2'));

    $output = $crud->render();

    $this->_example_output($output);
}

function add_field_callback_1()
{
    return '+30 <input type="text" maxlength="50" value="" name="phone"
style="width:462px">';
}

function add_field_callback_2()
{
    return '<input type="text" maxlength="50" value="" name="state"
style="width:400px"> ( for U.S. only )';
}
```

callback_after_delete

void callback_after_delete(mixed \$callback)

The callback runs when the operation delete completed successfully. The parameter that is insert is the primary key value. A return value is not required for this callback.

Example:

```
public function user(){
    $crud = new grocery_CRUD();

    $crud->set_table('cms_user');
    $crud->set_subject('User List');
    $crud->required_fields('username');

    $crud->columns('username','email','real_name','active');
    $crud->change_field_type('active', 'true_false');

    $crud->callback_after_delete(array($this,'user_after_delete'));

    $output = $crud->render();

    $this->_example_output($output);
}

public function user_after_delete($primary_key)
{
    return $this->db->insert('user_logs',array('user_id' =>
$primary_key,'action'=>'delete', 'updated' => date('Y-m-d H:i:s')));
}
```


callback_after_insert

```
void callback_after_insert(mixed $callback )
```

This is a callback after the auto insert of the CRUD. The function of the callback takes two parameters , 1 - the post data and 2 - the insert key value . Return value for this callback is not required.

Example:

```
public function users() {
    $crud = new grocery_CRUD();

    $crud->set_table('users');
    $crud->set_subject('User');
    $crud->required_fields('username');

    $crud->columns('username', 'email', 'real_name', 'active');
    $crud->fields('username', 'email', 'password', 'real_name', 'active');

    $crud->callback_after_insert(array($this, 'log_user_after_insert'));
    $crud->callback_after_update(array($this, 'log_user_after_update'));

    $output = $crud->render();

    $this->_example_output($output);
}

function log_user_after_insert($post_array,$primary_key)
{
    $user_logs_insert = array(
        "user_id" => $primary_key,
        "created" => date('Y-m-d H:i:s'),
        "last_update" => date('Y-m-d H:i:s')
    );

    $this->db->insert('user_logs',$user_logs_insert);

    return true;
}
```

callback_after_update

```
void callback_after_update(mixed $callback )
```

This is a callback that is used after the automatic update of the CRUD. The function of the callback takes two parameters , 1 - the post data and 2 - the primary key value . Return value for this callback is not required.

Example:

```
public function users() {
    $crud = new grocery_CRUD();

    $crud->set_table('users');
    $crud->set_subject('User');
    $crud->required_fields('username');

    $crud->columns('username', 'email', 'real_name', 'active');
    $crud->fields('username', 'email', 'password', 'real_name', 'active');

    $crud->callback_after_insert(array($this, 'log_user_after_insert'));
    $crud->callback_after_update(array($this, 'log_user_after_update'));

    $output = $crud->render();

    $this->_example_output($output);
}

function log_user_after_update($post_array,$primary_key)
{
    $user_logs_update = array(
        "user_id" => $primary_key,
        "last_update" => date('Y-m-d H:i:s')
    );

    $this->db->update('user_logs',$user_logs_update,array('user_id' =>
    $primary_key));

    return true;
}
```

callback_after_upload

```
void callback_after_upload( mixed $callback )
```

Available for version **>= 1.2**

A callback that triggered after the upload functionality. This is very useful if you want to resize the file, create thumbnail, zip it e.t.c.

I have an example with a simple resizing using [image moo](#) for the callback_after_upload

```
function employees_management()
{
    $crud = new grocery_CRUD();

    $this->load->config('grocery_crud');
    $this->config->set_item('grocery_crud_file_upload_allow_file_types',
                                                                    'gif|jpeg|jpg|png');

    $crud->set_table('employees');
    $crud->set_relation('officeCode','offices','city');
    $crud->display_as('officeCode','Office City');
    $crud->set_subject('Employee');

    $crud->set_field_upload('file_url','assets/uploads/images');

    $crud->callback_after_upload(array($this,'example_callback_after_upload'));

    $output = $crud->render();

    $this->_example_output($output);
}

/*
 * Examples of what the $uploader_response, $files_to_upload and $field_info will
 be:
 $uploader_response = Array
 (
     [0] => stdClass Object
     (
         [name] => 6d9c1-52.jpg
         [size] => 495375
         [type] => image/jpeg
         [url] => http://grocery_crud/assets/uploads/files/6d9c1-52.jpg
     )
 )
```

```

    )

$field_info = stdClass Object
(
    [field_name] => file_url
    [upload_path] => assets/uploads/files
    [encrypted_field_name] => sd1e6fec1
)

$files_to_upload = Array
(
    [sd1e6fec1] => Array
        (
            [name] => 86.jpg
            [type] => image/jpeg
            [tmp_name] => C:\wamp\tmp\phpFC42.tmp
            [error] => 0
            [size] => 258177
        )
    )
)
*/
function example_callback_after_upload($uploader_response,$field_info,
$files_to_upload)
{
    $this->load->library('image_moo');

    //Is only one file uploaded so it ok to use it with $uploader_response[0].
    $file_uploaded = $field_info->upload_path.'/'.$uploader_response[0]->name;

    $this->image_moo->load($file_uploaded)->resize(800,600)-
>save($file_uploaded,true);

    return true;
}

```

callback_before_delete

void callback_before_delete(mixed \$callback)

This callback runs before the auto delete of the crud. It takes one parameter - the primary key value. The return value is not required for this callback.

Example:

```
public function user(){
    $crud = new grocery_CRUD();

    $crud->set_table('cms_user');
    $crud->set_subject('User List');
    $crud->required_fields('username');

    $crud->columns('username','email','real_name','active');
    $crud->change_field_type('active', 'true_false');

    $crud->callback_before_delete(array($this,'log_user_before_delete'));

    $output = $crud->render();

    $this->_example_output($output);
}

public function log_user_before_delete($primary_key)
{
    $this->db->where('id',$primary_key);
    $user = $this->db->get('cms_user')->row();

    if(empty($user))
        return false;

    $this->db->insert('user_logs',array(
        'user_id' => $primary_key,
        'action'=>'delete',
        'email' => $user->email
        'updated' => date('Y-m-d H:i:s')));

    return true;
}
```

callback_before_insert

```
void callback_before_insert(mixed $callback )
```

This callback runs before the auto insert of the crud. The callback takes as a 1st parameter the post array. The return value is not required. Although if you return a value must be an array like a post array. With this opportunity you can add or unset some post variables.

Example:

```
public function users(){
    $crud = new grocery_CRUD();
    $crud->set_table('db_user');
    $crud->set_subject('User');
    $crud->required_fields('user_name');

    $crud->columns('user_name','email','real_name','active','groups');
    $crud->fields('user_name','email','password','real_name','active','groups');
    $crud->change_field_type('password','password');

    $crud->callback_before_insert(array($this,'encrypt_password_callback'));

    $output = $crud->render();
    $this->_example_output($output);
}

function encrypt_password_callback($post_array) {
    $this->load->library('encrypt');
    $key = 'super-secret-key';
    $post_array['password'] = $this->encrypt->encode($post_array['password'],
    $key);

    return $post_array;
}
```

Note: Be careful when there are fields that are not included at the fields method then you cannot just add more fields at the callback_before_insert. For example at the current example if we have something like that:

```
function encrypt_password_callback($post_array) {
    $this->load->library('encrypt');
    $key = 'super-secret-key';
    $post_array['password'] = $this->encrypt->encode($post_array['password'],
    $key);
}
```

```
$post_array['insert_date'] = date('Y-d-m');  
$post_array['updated_date'] = date('Y-d-m');  
  
return $post_array;  
}
```

it will NOT work as for security reasons grocery CRUD doesn't recognize the insert_date and the update_date field. To make it recognized you have to add it at the fields method (fields, add_fields or edit_fields). So for the current example we will have:

```
$crud->  
>fields('insert_date','updated_date','user_name','email','password','real_name','  
active','groups');
```

And of course after this we have to remove the fields from the form so we can transform these two fields to invisible fields. You can simply do that in the current example like this:

```
$crud->change_field_type('insert_date','invisible');  
$crud->change_field_type('updated_date','invisible');
```

For more about "invisible" fields you can read the [invisible field type](#)

callback_before_update

```
void callback_before_update(mixed $callback )
```

This callback runs before the auto update of the crud. The callback takes as a 1st parameter the post array and as 2nd the primary key value. The return value is not required. Although if you return a value must be an array like a post array. With this opportunity you can add or unset some post variables.

Example:

```
public function users(){
    $crud = new grocery_CRUD();
    $crud->set_table('db_user');
    $crud->set_subject('User');
    $crud->required_fields('user_name');

    $crud->columns('user_name','email','real_name','active','groups');
    $crud->fields('user_name','email','password','real_name','active','groups');

    $crud-
>callback_edit_field('password',array($this,'set_password_input_to_empty'));
    $crud-
>callback_add_field('password',array($this,'set_password_input_to_empty'));

    $crud->callback_before_update(array($this,'encrypt_password_callback'));

    $output = $crud->render();
    $this->_example_output($output);
}

function encrypt_password_callback($post_array, $primary_key) {
    $this->load->library('encrypt');

    //Encrypt password only if is not empty. Else don't change the password to an
    empty field
    if(!empty($post_array['password']))
    {
        $key = 'super-secret-key';
        $post_array['password'] = $this->encrypt->encode($post_array['password'],
$key);
    }
    else
    {
        unset($post_array['password']);
    }
}
```



```
    return $post_array;
}

function set_password_input_to_empty() {
    return "<input type='password' name='password' value='' />";
}
```

callback_before_upload

```
void callback_before_upload( mixed $callback )
```

Available for version **>= 1.2**

A callback that triggered before the upload functionality. This callback is suggested for validation checks.

If you return a string then it will alert the message of the string before the upload. So the upload will not work and the message will be alerted to the end-user. If you return a simple *false* then the default message of the upload error will appear, without of course uploading the file. If you return *true*, the upload will continue as normal.

You can see a full example of `callback_before_upload` below:

```
<?php
```

```
function employees_management()
{
    $crud = new grocery_CRUD();

    $crud->set_table('employees');
    $crud->set_relation('officeCode', 'offices', 'city');
    $crud->display_as('officeCode', 'Office City');
    $crud->set_subject('Employee');

    $crud->set_field_upload('file_url', 'assets/uploads/files');
    $crud->callback_before_upload(array($this, 'example_callback_before_upload'));

    $output = $crud->render();

    $this->_example_output($output);
}

function example_callback_before_upload($files_to_upload, $field_info)
{
    /*
     * Examples of what the $files_to_upload and $field_info will be:
     * $files_to_upload = Array
     * (
     *     [sd1e6fec1] => Array
     *     (
     *         [name] => 86.jpg
     *         [type] => image/jpeg
     */
}
```

```

        [tmp_name] => C:\wamp\tmp\phpFC42.tmp
        [error] => 0
        [size] => 258177
    )
)

$field_info = stdClass Object
(
    [field_name] => file_url
    [upload_path] => assets/uploads/files
    [encrypted_field_name] => sd1e6fec1
)

*/

if(is_dir($field_info->upload_path))
{
    return true;
}
else
{
    return 'I am sorry but it seems that the folder that you are trying to
upload doesn\'t exist.';
}
}

```

callback_column

```
void callback_column( string $column , mixed $callback )
```

This callback runs on each row. It escapes the auto column value and runs the callback. For this callback the return value is required and must be a string. The parameters that callback takes are : 1 - the primary key value of the row and 2 - the row as an object. The row as an object we can use it if we want quickly to take a value for another field.

Example:

```
public function webpages()
{
    $c = new grocery_CRUD();

    $c->set_table('webpages');
    $c->order_by('priority');
    $c->set_subject('Webpage');
    $c->columns('menu_title','url','status','priority');

    $c->callback_column('menu_title',array($this,'_callback_webpage_url'));

    $output = $c->render();
    $this->_view_output($output);
}

public function _callback_webpage_url($value, $row)
{
    return "<a href='".site_url('admin/sub_webpages/'.$row->id)."'>$value</a>";
}
```

callback_delete

```
void callback_delete(mixed $callback )
```

This callback escapes the auto delete of the CRUD , and runs only the callback.
Below you can see an example:

```
public function user(){
    $crud = new grocery_CRUD();

    $crud->set_table('cms_user');
    $crud->set_subject('User List');
    $crud->required_fields('user_name');

    $crud->columns('user_name','email','real_name','active');
    $crud->change_field_type('active','true_false');

    $crud->callback_delete(array($this,'delete_user'));

    $output = $crud->render();

    $this->_example_output($output);
}

public function delete_user($primary_key)
{
    return $this->db->update('cms_user',array('deleted' => '1'),array('id' =>
$primary_key));
}
```

callback_edit_field

```
void callback_edit_field( string $field , mixed $callback )
```

This callback escapes the default auto field output of the field name at the edit form. To your callback you will take two parameters . 1st the value of the field and 2nd the primary key value of the record you just edited.

Example:

```
function example_callback_edit_field(){
    $crud = new grocery_CRUD();

    $crud->set_table('offices');
    $crud->set_subject('Office');
    $crud->required_fields('city');
    $crud->columns('city','country','phone','addressLine1','postalCode');

    $crud->callback_edit_field('phone',array($this,'edit_field_callback_1'));

    $output = $crud->render();

    $this->_example_output($output);
}

function edit_field_callback_1($value, $primary_key)
{
    return '+30 <input type="text" maxlength="50" value="'.$value.'" name="phone"
style="width:462px">';
}
```

callback_field

```
void callback_field( string $field ,mixed $callback )
```

callback_field is just a shortcut to callback_add_field and callback_edit_field.

So for example if you have:

```
$crud->callback_field('phone',array($this,'example_callback'));
```

is similar to:

```
$crud->callback_add_field('phone',array($this,'example_callback'));  
$crud->callback_edit_field('phone',array($this,'example_callback'));
```

Example:

```
function example_callback_field(){  
    $crud = new grocery_CRUD();  
  
    $crud->set_table('offices');  
    $crud->set_subject('Office');  
    $crud->required_fields('city');  
    $crud->columns('city','country','phone','addressLine1','postalCode');  
  
    $crud->callback_field('phone',array($this,'field_callback_1'));  
  
    $output = $crud->render();  
  
    $this->_example_output($output);  
}  
  
function field_callback_1($value = '', $primary_key = null)  
{  
    return '+30 <input type="text" maxlength="50" value="'. $value .'" name="phone" style="width:462px">';  
}
```

callback_insert

```
void callback_insert(mixed $callback )
```

This callback escapes the auto insert of the CRUD, and runs only the inserted callback.

Example:

```
public function users(){
    $crud = new grocery_CRUD();
    $crud->set_table('db_user');
    $crud->set_subject('User');
    $crud->required_fields('user_name');

    $crud->columns('user_name','email','real_name','active','groups');
    $crud->fields('user_name','email','password','real_name','active','groups');
    $crud->change_field_type('password','password');

    $crud->callback_insert(array($this,'encrypt_password_and_insert_callback'));

    $output = $crud->render();
    $this->_example_output($output);
}

function encrypt_password_and_insert_callback($post_array) {
    $this->load->library('encrypt');
    $key = 'super-secret-key';
    $post_array['password'] = $this->encrypt->encode($post_array['password'],
    $key);

    return $this->db->insert('db_user',$post_array);
}
```


callback_update

```
void callback_update(mixed $callback )
```

This callback escapes the auto update of the CRUD , and runs only the callback.

Example:

```
public function users(){
    $crud = new grocery_CRUD();
    $crud->set_table('db_user');
    $crud->set_subject('User');
    $crud->required_fields('user_name');

    $crud->columns('user_name','email','real_name','active','groups');
    $crud->fields('user_name','email','password','real_name','active','groups');

    $crud-
>callback_edit_field('password',array($this,'set_password_input_to_empty'));
    $crud-
>callback_add_field('password',array($this,'set_password_input_to_empty'));

    $crud->callback_update(array($this,'encrypt_password_and_update'));

    $output = $crud->render();
    $this->_example_output($output);
}

function encrypt_password_and_update($post_array, $primary_key) {
    $this->load->library('encrypt');

    //Encrypt password only if is not empty. Else don't change the password to an
    empty field
    if(!empty($post_array['password']))
    {
        $key = 'super-secret-key';
        $post_array['password'] = $this->encrypt->encode($post_array['password'],
$key);
    }
    else
    {
        unset($post_array['password']);
    }

    return $this->db->update('db_user',$post_array,array('id' => $primary_key));
}
```

```
function set_password_input_to_empty() {  
    return "<input type='password' name='password' value='' />";  
}
```

callback_upload

```
void callback_upload( mixed $callback )
```

Available for version **>= 1.2**

A callback that replaces the default auto uploader.

change_field_type

```
void change_field_type( string $field , string $field_type [ , string $value ] )
```

This method has been deprecated and it is highly recommended to use field type instead that works with the exact same way .

For example when we have:

```
$crud->field_type('field_name', 'password');
```

it is exactly the same thing with:

```
$crud->change_field_type('field_name', 'password');
```

columns

```
void columns( string $var [, string $var [, string $... ]] )
```

The displayed columns that user see

Example:

```
$crud->columns('customerName','phone','addressLine1','creditLimit');
```

or else:

```
$crud->columns(array('customerName','phone','addressLine1','creditLimit'));
```

display_as

```
void display_as ( $field_name , $display_as )
```

Changes the displaying label of the field. The label will change the field label and the column label. You can add any utf8 character you like. For example you can translate all the fields to your language

Example:

```
function full_example()
{
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        -

>columns('customerName', 'contactLastName', 'phone', 'city', 'country', 'creditLimit')
;

    $crud->display_as('customerName', 'Name')->display_as('contactLastName', 'Last
Name');

    $crud-
>fields('customerName', 'contactLastName', 'phone', 'city', 'country', 'creditLimit');
    $crud->required_fields('customerName', 'contactLastName');

    $output = $crud->render();

    $this->_example_output($output);
}
```

edit_fields

```
void edit_fields( string $var [, string $var [, string $... ]] )
```

The fields that user will see on edit operation.

Example:

```
function customers_example() {
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        ->columns('customerName','contactLastName','creditLimit');

    $crud->add_fields('customerName','contactLastName','city','creditLimit');
    $crud->edit_fields('customerName','contactLastName','city');

    $crud->required_fields('customerName','contactLastName');

    $output = $crud->render();

    $this->_example_output($output);
}
```

fields

```
void fields( string $var [, string $var [, string $... ]] )
```

The fields that user will see on add and edit form

Example:

```
function full_example()
{
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        -
    >columns('customerName','contactLastName','phone','city','country','creditLimit')
        ->display_as('customerName','Name')
        ->display_as('contactLastName','Last Name');

    $crud-
    >fields('customerName','contactLastName','phone','city','country','creditLimit');
    $crud->required_fields('customerName','contactLastName');

    $output = $crud->render();

    $this->_example_output($output);
}
```


field_type

```
void field_type( string $field , string $field_type [ , string $value ] )
```

Available for version >= 1.2.3

Changes the default field type.

The field type is a string and can take the following options:

- hidden
- invisible
- password
- enum
- set
- dropdown
- multiselect
- integer
- true_false
- string
- text
- date
- datetime
- readonly

Note: The third parameter (\$value) only works for the hidden, enum and set field and it is not a default value for all the other types.

Hidden field

An example of how to use hidden field:

```
function hidden_test($office_id = 0)
{
    $crud = new grocery_CRUD();

    $crud->set_table('customers');
    $crud->
>columns('customerName','contactLastName','phone','city','country','salesRepEmployeeNumber','creditLimit');
    $crud->display_as('salesRepEmployeeNumber','from Employee');
    $crud->set_subject('Customer');
    $crud->set_relation('salesRepEmployeeNumber','employees','lastName');
    $crud->
>add_fields('customerName','contactLastName','phone','city','country','salesRepEmployeeNumber','creditLimit','office_id');
```

```

    $crud-
>edit_fields('customerName','contactLastName','phone','city','country','salesRepEmployeeNumber','creditLimit');

    $crud->field_type('office_id', 'hidden', $office_id);

    $output = $crud->render();

    $this->_example_output($output);
}

```

With this code you can see that office_id will disappear from the add and edit form. The office_id will be a hidden field with value of \$office_id . You can of course add a static value, for example:

```

$crud->field_type('office_id', 'hidden', 3);

```

As you can see the office_id is ONLY in the \$crud->add_fields and not \$crud->edit_fields. This is just to understand that even if we write the field_type method , you must add it to your fields (add_fields / edit_fields). The most common is to use the method fields that is for add form and edit form. In this case it would be:

```

$crud-
>fields('customerName','contactLastName','phone','city','country','salesRepEmployeeNumber','creditLimit','office_id');

```

Invisible Field

Many people are confused about how and why to use "invisible" fields. So for example if you have:

```

$crud->fields('field1','field2','field3');
$crud->callback_before_insert(array($this,'test_callback'));

```

and :

```

function test_callback($post_array){
    $post_array['field4'] = 'test';
    return $post_array;
}

```

This will NOT work as expected unless you add the invisible field so you have to do it like this:

```
$crud->fields('field1','field2','field3','field4');
$crud->field_type('field4','invisible');
$crud->callback_before_insert(array($this,'test_callback'));
```

So this WILL WORK as you expected and without showing the field "field4" in the form. The "invisible" field type is created for security reasons

Password field

The password field just transforms the input type name to input type password, nothing more. To use it, you just add this line of code:

```
$crud->field_type('field_name', 'password');
```

and your input appears to the add and edit form as type = password

Below we can see a full example for using the password field for encrypt and decrypting password:

```
public function users(){
    $crud = new grocery_CRUD();

    $crud->set_table('db_user');
    $crud->set_subject('User');
    $crud->required_fields('user_name');
    $crud->columns('user_name','email','real_name','active', 'groups');
    $crud->fields('user_name','email','password','real_name','active', 'groups');

    $crud->field_type('password', 'password');

    $crud->callback_before_insert(array($this,'encrypt_password_callback'));
    $crud->callback_before_update(array($this,'encrypt_password_callback'));
    $crud->callback_edit_field('password',array($this,'decrypt_password_callback'));

    $output = $crud->render();
    $this->_example_output($output);
}

function encrypt_password_callback($post_array, $primary_key = null)
{
    $this->load->library('encrypt');
```

```

        $key = 'super-secret-key';
        $post_array['password'] = $this->encrypt->encode($post_array['password'],
$key);
        return $post_array;
    }

function decrypt_password_callback($value)
{
    $this->load->library('encrypt');

    $key = 'super-secret-key';
    $decrypted_password = $this->encrypt->decode($value, $key);
    return "<input type='password' name='password' value='$decrypted_password'
/>";
}

```

Enum field

Available for version >= 1.2.3

```
$crud->field_type('status','enum',array('active','private','spam','deleted'));
```

Set field

Available for version >= 1.2.3

```
$crud->field_type('fruits','set',array('banana','orange','apple','lemon'));
```

Dropdown field

Available for version >= 1.3.2

```
$crud->field_type('status','dropdown',
    array('1' => 'active', '2' => 'private', '3' => 'spam' , '4' =>
'deleted'));

```

Multiselect field

Available for version >= 1.3.2

```
$crud->field_type('fruits','multiselect',
    array( "1"  => "banana", "2" => "orange", "3" =>
"apple"));
```

getState

```
string getState()
```

We can see an example of how it works below:

Let's say we have our function `employees_management`. I have created two functions named `getState` and `getStateInfo`. `getState` will return a value from the array:

```
$states = array(  
    0 => 'unknown',  
    1 => 'list',  
    2 => 'add',  
    3 => 'edit',  
    4 => 'delete',  
    5 => 'insert',  
    6 => 'update',  
    7 => 'ajax_list',  
    8 => 'ajax_list_info',  
    9 => 'insert_validation',  
    10 => 'update_validation',  
    11 => 'upload_file',  
    12 => 'delete_file',  
    13 => 'ajax_relation',  
    14 => 'ajax_relation_n_n',  
    15 => 'success',  
    16 => 'export',  
    17 => 'print'  
);
```

So if you need for example to handle the State 'add' and 'edit' you will do:

```
function employees_management()  
{  
    $crud = new grocery_CRUD();  
  
    $crud->set_theme('datatables');  
    $crud->set_table('employees');  
    $crud->set_relation('officeCode','offices','city');  
    $crud->display_as('officeCode','Office City');  
    $crud->set_subject('Employee');  
  
    $output = $crud->render();  
  
    $state = $crud->getState();  
    $state_info = $crud->getStateInfo();  
}
```

```
if($state == 'add')
{
    //Do your cool stuff here . You don't need any State info you are in add
}
elseif($state == 'edit')
{
    $primary_key = $state_info->primary_key;
    //Do your awesome coding here.
}
else
{
    $this->_example_output($output);
}
}
```

getStateInfo

```
object getStateInfo()
```

get_field_types

array get_field_types()

get_primary_key

```
string get_primary_key
```

like

```
void like( mixed $field [ , string $match [ , string $side ] ] )
```

It works exactly with the same way as Codeigniter's like.

For more you can also check [codeigniter active record](#)

Example:

```
function example_with_like() {

    $crud = new grocery_CRUD();

    $crud->like('productName', 'Motor');

    $crud->set_table('products');
    $crud->columns('productName', 'buyPrice');

    $output = $crud->render();

    $this->_example_output($output);
}
```

limit

```
void limit(mixed $limit [, mixed $offset = ''] )
```

order_by

```
void order_by(mixed $order_by [, string $direction] )
```

A quick ordering when the user first visits the list page(work the same way as codeigniter)

Example:

```
function customers_example() {
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        ->columns('customerName', 'contactLastName', 'creditLimit');

    $crud->fields('customerName', 'contactLastName', 'city', 'creditLimit');

    $crud->order_by('creditLimit', 'desc');

    $output = $crud->render();

    $this->_example_output($output);
}
```

or_like

```
void or_like( mixed $field [ , string $match [, string $side] ] )
```

It works exactly with the same way as Codeigniter's like.

For more you can also read [codeigniter active record](#) documentation

Example:

```
function example_with_or_like() {

    $crud = new grocery_CRUD();

    $crud->like('productName', 'Moto');
    $crud->or_like('productName', 'Car');
    $crud->or_like('productName', 'Bicycle');

    $crud->set_table('products');
    $crud->columns('productName', 'buyPrice');

    $output = $crud->render();

    $this->_example_output($output);
}
```

or_where

```
void or_where( mixed $key [, string $value [, bool $escape] ] )
```

It works exactly with the same way as Codeigniter's `or_where`.

For more you can also read [codeigniter active record](#) documentation

Example:

```
function example_with_or_where() {

    $crud = new grocery_CRUD();

    $crud->where('productName', 'Motorcycle');
    $crud->or_where('productName', 'Car');
    $crud->or_where('productName', 'Bicycle');

    $crud->set_table('products');
    $crud->columns('productName', 'buyPrice');

    $output = $crud->render();

    $this->_example_output($output);
}
```

render

```
void render()
```

Or else ... make it work!! The web application takes decision of what to do and show it to the user. This is the most important method in grocery CRUD. Without this method nothing happens.

So for example if we have:

```
$crud = new grocery_CRUD();

$crud->set_table('customers')
->set_subject('Customer')
-
>columns('customerName','contactLastName','phone','city','country','creditLimit')
->display_as('customerName','Name')
->display_as('contactLastName','Last Name');

$crud-
>fields('customerName','contactLastName','phone','city','country','creditLimit');
$crud->required_fields('customerName','contactLastName');
```

With the above code we just set the properties/features that our CRUD will have. And with the method render:

```
$output = $crud->render();
```

everything will work. You don't need to add anything else to make it work, just the render.

You can see a full example of using the render method below:

```
function full_example()
{
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        -
    >columns('customerName','contactLastName','phone','city','country','creditLimit')
        ->display_as('customerName','Name')
        ->display_as('contactLastName','Last Name');

    $crud-
    >fields('customerName','contactLastName','phone','city','country','creditLimit');
    $crud->required_fields('customerName','contactLastName');

    $output = $crud->render();

    $this->_example_output($output);
}
```

required_fields

`void required_fields(string $var [, string $var [, string $...]])`
Sets the required fields of add and edit fields.

```
function customers_management()  
{  
    $crud = new grocery_CRUD();  
  
    $crud->set_table('customers');  
    $crud->  
>columns('customerName','contactLastName','phone','city','country','salesRepEmployeeNumber','creditLimit');  
    $crud->display_as('salesRepEmployeeNumber','from Employeeer')  
        ->display_as('customerName','Name')  
        ->display_as('contactLastName','Last Name');  
  
    $crud->required_fields('customerName','contactLastName');  
  
    $crud->set_subject('Customer');  
    $crud->set_relation('salesRepEmployeeNumber','employees','lastName');  
  
    $output = $crud->render();  
  
    $this->_example_output($output);  
}
```


set_crud_url_path

```
void set_crud_url_path(string $crud_url_path [, string $list_url_path ])
```

Available for version **>= 1.4**

Set a full URL path to this method.

This method is useful when the path is not specified correctly. Especially when we are using routes. For example: Let's say we have the path `http://www.example.com/` however the original url path is `http://www.example.com/example/index` . We have to specify the url so we can have all the CRUD operations correctly. The url path has to be set from this method like this:

```
$crud->set_crud_url_path(site_url('example/index'));
```

set_field_upload

```
void set_field_upload( string $field_name, string $upload_path)
```

Sets a field name to be an uploaded file.

Note: You cannot insert the slash (/) at the beginning of your string. For example if your path is assets/uploads/files from the beginning of your project you cannot insert "/assets/uploads/files", because it will not work.

Example:

```
function employees_management()  
{  
    $crud = new grocery_CRUD();  
  
    $crud->set_theme('datatables');  
    $crud->set_table('employees');  
    $crud->set_relation('officeCode','offices','city');  
    $crud->display_as('officeCode','Office City');  
    $crud->set_subject('Employee');  
  
    $crud->required_fields('lastName');  
  
    $crud->set_field_upload('file_url','assets/uploads/files');  
  
    $output = $crud->render();  
  
    $this->_example_output($output);  
}
```

set_language

```
void set_language( string $language )
```

You can simply change your language at your method. So for example in your project if you have the default language to "english" you can have one situation that you don't want to have the default language but another, just type

```
$crud->set_language("greek").
```

And you will have English only at this method.

This functionality could help you create a multilingual site so each time you change a language you can have a type of code like this:

```
$crud->set_language($this->session->('language'));
```

set_lang_string

```
void set_lang_string( string $handle, string $lang_string )
```

Set your lang string directly. This is really useful if you want just for some cases to change the language string. Let's have an example:

```
$crud = new grocery_CRUD();
$crud->set_table('customers');
$crud->columns('customerName','phone','addressLine1','creditLimit');

$crud->set_lang_string('form_update_changes','Updating existing customer')
    ->set_lang_string('form_back_to_list','Go back to
customers page')
    ->set_lang_string('form_save','Save customer into the
database');

$output = $crud->render();

$this->_example_output($output);
```

set_model

```
void set_model( string $model_as_string );
```

Sets the model that crud will use .The model always must extends grocery_CRUD_Model .
grocery_CRUD_Model extends CI_Model so you don't have to be afraid to use it.

Below I explain with steps how you can use the set_model

1st STEP - go to your basic function and add your custom model

```
function just_an_example()
```

```
{
    $crud = new grocery_CRUD();

    $crud->set_model('My_Custom_model');
    $crud->set_table('film');
    $crud->set_relation_n_n('actors', 'film_actor', 'actor', 'film_id',
'actor_id', 'fullname', 'priority');

    $output = $crud->render();

    $this->_example_output($output);
}
```

2nd STEP-Your custom model MUST extend grocery_CRUD_Model. So create a new file at
application/models/my_custom_model.php and it will be something like this:

```
class My_Custom_model extends grocery_CRUD_Model {

    function get_relation_n_n_unselected_array($field_info, $selected_values)

    {
        $selection_primary_key = $this->get_primary_key($field_info->selection_table);

        if($field_info->name = '.....')

        {
            $this->db->where(...);
            .....your custom queries
        }

        $this->db->order_by("{ $field_info->selection_table }. { $field_info->title_field_selection_table }");

        $results = $this->db->get($field_info->selection_table)->result();
    }
}
```

```
    $results_array = array();
    foreach($results as $row)

    {
        if(!isset($selected_values[$row->{$field_info-
>primary_key_alias_to_selection_table}])))
            $results_array[$row->{$field_info-
>primary_key_alias_to_selection_table}] = $row->{$field_info-
>title_field_selection_table};
        }

    return $results_array;
}

}
```

set_primary_key

```
void set_primary_key( string $primary_key_field [ , string $table_name ] )
```

Available for version **>= 1.2.3**

Handles the default primary key for a specific table. If the \$table_name is NULL then the primary key is for the default table name that we added at the set_table method. Really useful for the mysql VIEW tables.

At the below example we have the table countries that has the below fields:

- country_id (primary key)
- country_code
- country_title

Let's say that we don't want to relate our table with the country_id but with the country_code as it also unique. At this situation we can do something similar to this:

```
function employees_management()  
{  
    $crud = new grocery_CRUD();  
  
    $crud->set_table('employees');  
    $crud->set_subject('Employee');  
  
    $crud->set_primary_key('country_code', 'countries');  
    $crud->set_relation('country', 'countries', 'country_title');  
  
    $output = $crud->render();  
  
    $this->_example_output($output);  
}
```

With the above example at the *country* field will be stored the *country_code* and **not** the *country_id*. At this example the:

```
$crud->set_primary_key('country_code', 'countries');
```

simply means that when you request the primary key from the table countries it will give you the country_code instead.

set_relation

```
void set_relation( string $field_name , string $related_table, string  
$related_title_field [, mixed $where [, string $order_by ] ] )
```

Set a relation 1-n database relation. This will automatically create a dropdown list to the fields and show the actual name of the field and not just a primary key to the list. An example of this:

```
$crud->set_relation('user_id','users','username');
```

You can have as many fields you like to call from the other table and the syntax is really simple. Just at the 3rd field you will have the symbol { and } . So it will be for example:

```
$crud->set_relation('user_id','users','{username} - {last_name} {first_name}');
```

And you can have whatever syntax or symbols you like. So for example you can have:

```
$crud->set_relation('user_id','users','{username} ( {last_name} {first_name} )');
```

The parenthesis is just to show you that you can insert whatever symbol you like.

You can also have a where clause at the 4th parameter (is not a required parameter). For example:

```
$crud->set_relation('user_id','users','username',array('status' => 'active'));
```

It works exactly like codeigniter's where clause (you can add an array or a string), with the only difference that you cannot add a 3rd parameter (for example true or false).

The 5th parameter (not required) is the order_by. For example:

```
$crud->set_relation('user_id','users','username',null,'priority ASC');
```

I added the 4th parameter as null because we don't need a where clause in this example. If we also need a where clause we can simply do:

```
$crud->set_relation('user_id','users','username',array('status' =>  
'active'),'priority ASC');
```

You can also see a simple working example below:

```
function employees_management()  
{  
    $crud = new grocery_CRUD();
```

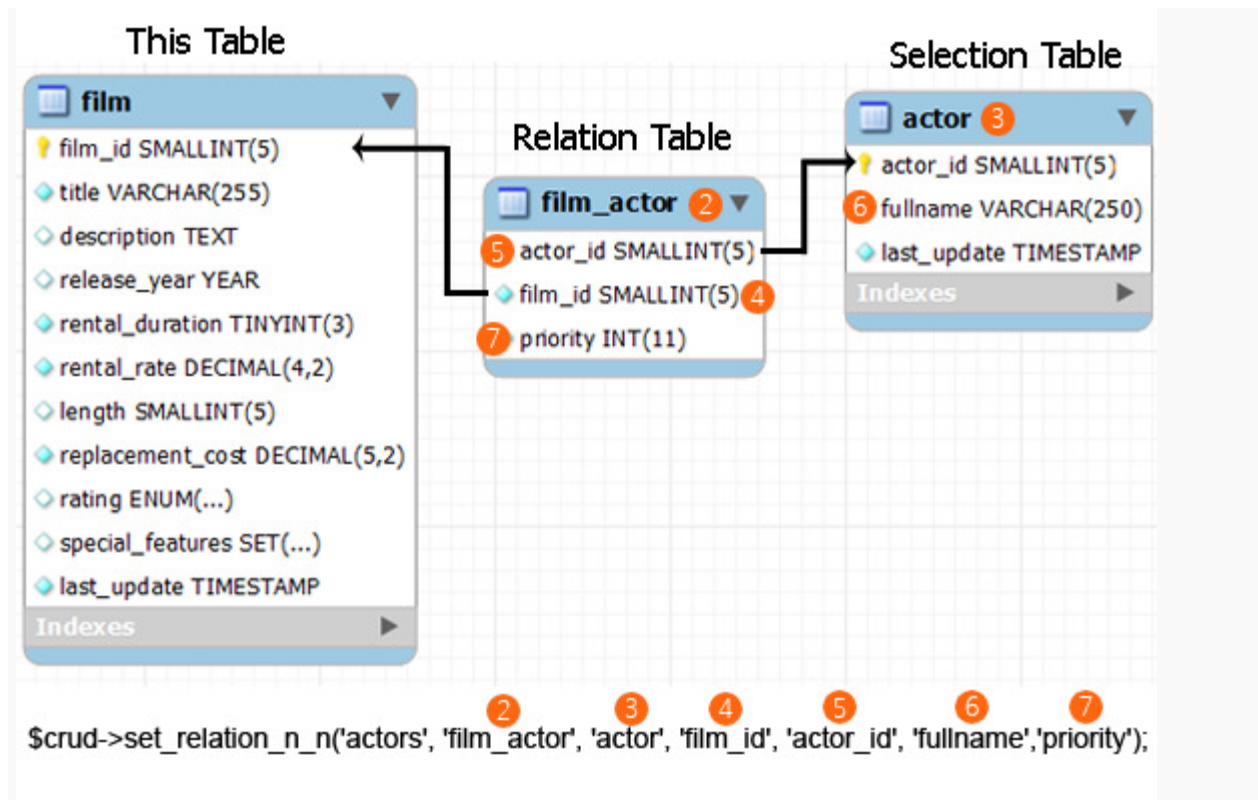


```
$crud->set_theme('datatables');  
$crud->set_table('employees');  
$crud->display_as('officeCode', 'Office City');  
$crud->set_subject('Employee');  
  
$crud->set_relation('officeCode', 'offices', 'city');  
  
$output = $crud->render();  
  
$this->_example_output($output);  
}
```

set_relation_n_n

```
void set_relation_n_n( string $field_name, string $relation_table, string  
$selection_table, string $primary_key_alias_to_this_table, string  
$primary_key_alias_to_selection_table , string $title_field_selection_table [ ,  
string $priority_field_relation ] )
```

Sets a relation with n-n relationship. There must include 3 tables. The basic table, the selection table and the relation table (or else middle table).



set_rules

```
void set_rules(mixed $field [, string $label [, string $rules] ])
```

Set Validation Rules works with the exact same way as Codeigniter set_rules. For more you can read: [codeigniter validation rules](#)

Important note: If the \$field is an array then no automated crud fields will take apart

```
function example_set_rules() {  
  
    $crud = new grocery_CRUD();  
  
    $crud->set_table('products');  
    $crud->columns('productName', 'buyPrice');  
  
    $crud->set_rules('buyPrice', 'buy Price', 'numeric');  
    $crud->set_rules('quantityInStock', 'Quantity In Stock', 'integer');  
  
    $output = $crud->render();  
  
    $this->_example_output($output);  
}
```

set_subject

```
void set_subject( string $subject )
```

This method is really useful when you want to specify what is the actual subject of your table CRUD. The default value is "record" and it is also translatable. It is very easy to set a subject and then this string is reused in almost every message or operation.

For example if we insert as a subject the string "Office" it will be:

- instead of "New record" we will have "New Office"
- instead of "Edit record" we will have "Edit Office"
- instead of "Are you sure that you want to remove this record?" we will have "Are you sure that you want to remove this "Office"
- and so on...

Below we can see a live example of how to use it:

```
function offices_management()  
{  
    $crud = new grocery_CRUD();  
  
    $crud->set_table('offices');  
    $crud->set_theme('datatables');  
    $crud->required_fields('city');  
    $crud->columns('city','country','phone','addressLine1','postalCode');  
  
    $crud->set_subject('Office');  
  
    $output = $crud->render();  
  
    $this->_example_output($output);  
}
```

set_table

```
void set_table(string $table_name)
```

Sets the basic database table that we will get our data.

Example:

```
function offices_management()  
{  
    $crud = new grocery_CRUD();  
  
    $crud->set_table('offices');  
  
    $crud->set_theme('datatables');  
    $crud->set_subject('Office');  
    $crud->required_fields('city');  
    $crud->columns('city', 'country', 'phone', 'addressLine1', 'postalCode');  
  
    $output = $crud->render();  
  
    $this->_example_output($output);  
}
```

set_theme

```
void set_theme(string $theme)
```

Set the CRUD theme - For now on there is only 'flexigrid' and 'datatables' . The default theme is flexigrid.

Note: flexigrid for now on is a server scripting grid and datatables is a client scripting grid.

Example:

```
function offices_management()
{
    $crud = new grocery_CRUD();

    $crud->set_theme('datatables');

    $crud->set_table('offices');
    $crud->set_subject('Office');
    $crud->required_fields('city');
    $crud->columns('city','country','phone','addressLine1','postalCode');

    $output = $crud->render();

    $this->_example_output($output);
}
```

unset_add

void unset_add()

Unsets the add operation - A user cannot add or insert records from the CRUD to database

Example:

```
function employees_delete_management()
{
    $crud = new grocery_CRUD();

    $crud->set_theme('datatables');
    $crud->set_table('employees');
    $crud->set_relation('officeCode','offices','city');
    $crud->display_as('officeCode','Office City');
    $crud->set_subject('Employee');
    $crud->unset_add();
    $crud->unset_edit();

    $output = $crud->render();
    $this->_example_output($output);
}
```

unset_add_fields

```
void unset_add_fields( string $var [, string $var [, string $... ]] )
```

Available for version **>= 1.2.1**

Unsets the fields at the add form.

Example:

```
$crud->unset_add_fields('name','address','postcode');
```

or similar to this:

```
$crud->unset_add_fields(array('name','address','postcode'));
```


unset_back_to_list

```
void unset_back_to_list()
```

Unset all the "back to list" buttons and messages.

Example:

```
function unset_back_to_list_example()
{
    $crud = new grocery_CRUD();

    $crud->set_theme('datatables');
    $crud->set_table('employees');
    $crud->set_relation('officeCode','offices','city');
    $crud->display_as('officeCode','Office City');
    $crud->set_subject('Employee');

    $crud->unset_back_to_list();

    $output = $crud->render();
    $this->_example_output($output);
}
```

unset_columns

```
void unset_columns( string $var [, string $var [, string $... ]] )
```

Unset columns from the list. This function is really useful when you want to show all the fields of the table to the list... except some fields

Example:

```
function film_management_list()
{
    $crud = new grocery_CRUD();

    $crud->set_table('film');
    $crud->set_theme('datatables');
    $crud->unset_operations();

    $crud->
>unset_columns('description','special_features','last_update','actors','category'
);

    $output = $crud->render();

    $this->_example_output($output);
}
```

unset_delete

```
void unset_delete()
```

Remove the delete operation from the CRUD list- A user cannot delete a record from the CRUD

Example:

```
function employees_unset_delete_test()
{
    $crud = new grocery_CRUD();

    $crud->set_table('employees');
    $crud->set_relation('officeCode', 'offices', 'city');
    $crud->display_as('officeCode', 'Office City');
    $crud->set_subject('Employee');

    $crud->unset_delete();

    $output = $crud->render();
    $this->_example_output($output);
}
```

unset_edit

```
void unset_edit()
```

Unsets the edit operation - A user cannot edit or update records from the CRUD

Example:

```
function employees_delete_management()
{
    $crud = new grocery_CRUD();

    $crud->set_theme('datatables');
    $crud->set_table('employees');
    $crud->set_relation('officeCode', 'offices', 'city');
    $crud->display_as('officeCode', 'Office City');
    $crud->set_subject('Employee');
    $crud->unset_add();
    $crud->unset_edit();

    $output = $crud->render();
    $this->_example_output($output);
}
```

unset_edit_fields

```
void unset_edit_fields( string $var [, string $var [, string $... ]] )
```

Available for version **>= 1.2.1**

unsets the fields at the edit form.

Example:

```
$crud->unset_edit_fields('name','address','postcode');
```

or else

```
$crud->unset_edit_fields(array('name','address','postcode'));
```

unset_export

```
void unset_export()
```

Available for version **>= 1.3**

Unset the export button and don't let the user to use this functionality.

Example:

```
function example_with_unset_export()
{
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        -
>columns('customerName','contactLastName','phone','city','country','creditLimit')
;

    $crud-
>fields('customerName','contactLastName','phone','city','country','creditLimit');
    $crud->required_fields('customerName','contactLastName');

    $crud->unset_export();

    $output = $crud->render();

    $this->_example_output($output);
}
```

unset_fields

```
void unset_fields( string $var [, string $var [, string $... ]] )
```

Available for version **>= 1.2.1**

Unset_fields is nothing more than a shortcut to **unset_add_fields** and **unset_edit_fields**
So for example if you have:

```
$crud->unset_fields('name','age');
```

it is just a shortcut to:

```
$crud->unset_add_fields('name','age');  
$crud->unset_edit_fields('name','age');
```

nothing more complicated than that.

You can also use the unset_fields with an array input so for example:

```
$crud->unset_fields(array('name','age'));
```

is the same thing as this:

```
$crud->unset_fields('name','age');
```

unset_jquery_ui

```
void unset_jquery_ui()
```

Available for version **>= 1.3.2**

This method is really useful when you already have loaded the jqueryUI from your default template.

Example:

```
$crud->unset_jquery_ui();
```


unset_list

```
void unset_list()
```

This functionality simply unset the list (datagrid). So the user is not able to access the list page.

Example:

```
function unset_list_example()
{
    $crud = new grocery_CRUD();

    $crud->set_theme('datatables');
    $crud->set_table('employees');
    $crud->set_relation('officeCode','offices','city');
    $crud->display_as('officeCode','Office City');
    $crud->set_subject('Employee');

    $crud->unset_list();

    $output = $crud->render();
    $this->_example_output($output);
}
```

Notice: When a user land to the first list webpage then an unhandle exception with message "You don't have permissions for this operation".

We didn't want to add a show_error there because then you cannot handle the exception. So a quick way to have a default Codeigniter error message is:

```
try{
    $crud->render();
} catch(Exception $e) {
    show_error($e->getMessage());
}
```

or if you want to do it with the right way and have a message for the user you can do it like this:

```
try{
    $crud->render();
} catch(Exception $e) {
    if($e->getCode() == 14) //The 14 is the code of the "You don't have permissions" error on grocery CRUD.
    {
        $this->load->view('permission_denied.php');//This is a custom view that you have to create
        //Or you can simply have an error message for this
        //For example: show_error('You don\'t have permissions for this operation');
    }
}
```

```
else
{
    show_error($e->getMessage());
}
}
```

And if you want the user to have for example the add form you can easily copy the url of the add form for example: localhost/your_project/index.php/examples/test/add and simply direct him there. Or you can also do this:

```
try{
    $crud->render();
} catch(Exception $e) {

    if($e->getCode() == 14) //The 14 is the code of the "You don't have permissions" error on grocery CRUD.
    {
        redirect(strtolower(__CLASS__).'/'.$e->strtolower(__FUNCTION__).'/'add');
    }
    else
    {
        show_error($e->getMessage());
    }
}
```

You have the freedom to do whatever you like rather than have it into the core library.

unset_operations

```
void unset_operations()
```

Unset all the operations . A user cannot insert, update or delete from the CRUD. User has access only to the table grid.

Example:

```
function film_management_list()
{
    $crud = new grocery_CRUD();

    $crud->set_table('film');
    $crud->set_theme('datatables');
    $crud->unset_columns('description','special_features','last_update','actors','category');

    $crud->unset_operations();

    $output = $crud->render();

    $this->_example_output($output);
}
```

unset_print

```
void unset_print()
```

Available for version **>= 1.3**

Unset the print button and don't let the user to use this functionality.

Example:

```
function example_with_unset_print()
{
    $crud = new grocery_CRUD();

    $crud->set_table('customers')
        ->set_subject('Customer')
        -
>columns('customerName','contactLastName','phone','city','country','creditLimit')
;

    $crud-
>fields('customerName','contactLastName','phone','city','country','creditLimit');
    $crud->required_fields('customerName','contactLastName');

    $crud->unset_print();

    $output = $crud->render();

    $this->_example_output($output);
}
```

unset_read

```
void unset_read()
```

Available for version **>= 1.4**

Unset's the read operation of the CRUD. In the CRUD is easy to unset the read operation by simply add this line of code:

```
$crud->unset_read();
```

unset_texteditor

```
void unset_texteditor(string $var [, string $var [, string $... ] ] )
```

Unsets the texteditor of the selected fields

Example:

```
$crud->unset_texteditor('description','full_text');
```

or else:

```
$crud->unset_texteditor(array('description','full_text'));
```

where

```
void where( mixed $key [, string $value [, bool $escape] ] )
```

A database where to our list. Works exactly the same way as codeigniter's where (\$this->db->where)

Example:

```
public function webpages()
{
    $crud = new grocery_CRUD();

    $crud->where('status','active');

    $crud->set_table('webpages');
    $crud->order_by('priority');
    $crud->set_subject('Webpage');
    $crud->columns('menu_title','url','status','priority');
    $crud->change_field_type('status', 'hidden', 'active');

    $output = $crud->render();
    $this->_view_output($output);
}
```