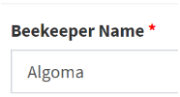
Validation

Validation is done using the codeigniter library **Form Validation**



Let's take the example of Beekeeper. The form views/beekeeper/edit.php is where the user complete the information about the Beekeeper. The required field are indicated with a red asterisk in the system. It means that a validation will be done to make sure that the field is filled.

Library Form Validation

It's in the controller beekeeper.php that the validation will be handle in the create and update functions.

```
$this->form_validation->set_rules('beekeeper_name', $this->lang->line('Beekeeper Name'), 'trim|required');
$this->form_validation->set_rules('category', $this->lang->line('Category'), 'trim|required');
$this->form_validation->set_rules('association', $this->lang->line('Association'), 'trim|required');
$this->form_validation->set_rules('nationality[]', $this->lang->line('Nationality'), 'trim|required');
$this->form_validation->set_rules('gender', $this->lang->line('Gender'), 'trim|required');
$this->form_validation->set_rules('address', $this->lang->line('Address'), 'trim|required');
$this->form_validation->set_rules('region', $this->lang->line('Region'), 'trim|required');
$this->form_validation->set_rules('province', $this->lang->line('Province'), 'trim|required');
$this->form_validation->set_rules('municipality', $this->lang->line('Municipality'), 'trim|required');
$this->form_validation->set_rules('district', $this->lang->line('District'), 'trim|required');
$this->form_validation->set_rules('birth_date', $this->lang->line('Birth Date'), 'trim|required|valid_date');
$this->form_validation->set_rules('education', $this->lang->line('Highest Educational Attainment'), 'trim|required');
$this->form_validation->set_rules('fund_source[]', $this->lang->line('Fund Source'), 'trim|required');
$this->form_validation->set_error_delimiters('<p class="alert alert-warning">','</p>');

if ($this->form_validation->run() == TRUE) {
    // True case, we create the new beekeeper
```

If the validation is completed, the update or create will be done. If not, the error message generated by the form_validation class will be sent to the view who will show the error message after the form have been submitted to the server.

```
<?php if($this->session->flashdata('success')): ?>
  <div class="alert alert-success alert-dismissible" role="alert">
    <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></button>
    <?php echo $this->session->flashdata('success'); ?>
  </div>
<?php elseif($this->session->flashdata('error')): ?>
    <div class="alert alert-error alert-dismissible" role="alert">
      <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-hidden="true">&times;</span></button>
      <?php echo $this->session->flashdata('error'); ?>
    </div>
<?php endif; ?>
```

Creating our own validation with callback method

The validation system supports callbacks to your own validation methods. This allows you to extend the validation class to meet your needs. For example, if you need to run a specific function or a database query to validate some information, you can create a callback method that does that. Let's create an example of this for the validation of dates. We need to verify the range of dates (date from and date to) and the rules of the validation greater_than works only for numbers, not for date.

I used the rules of form_validation to indicate that it's required but for the date_end, I created a function to check the date.

```
$this->form_validation->set_rules('date_issued', $this->lang->line('Date issued'), 'trim|required');

$this->form_validation->set_rules('date_end', $this->lang->line('Date end'), 'trim|required|callback_date_check');
```

```
public function date_check()
{
    if ($this->input->post('date_end') < $this->input->post('date_issued') )
        {$this->form_validation->set_message('date_check', $this->lang->line('The date End should be greater
            or equal to the date Issued.'));
        return FALSE;
        }
    else
        {return TRUE;}
}
```

The message should be translated in both language.

Check Integrity

More specific validation will be done for example, when deleting settings table that might be used in the system. In all the remove function of these tables, you will find this validation. In this example, the District can't be delete if the function checkIntegrity indicate that some rows are using the entry.

```
if($district_id) {
    //---> Validate if the information is used in beekeeper/apiary/association table
    $total_rows = $this->model_district->checkIntegrity($district_id);
    //---> If no beekeeper/apiary/association have this information, we can delete
    if ($total_rows == 0) {
        $delete = $this->model_district->remove($district_id);
        if($delete == true) {
            $response['success'] = true;
            $response['messages'] = $this->lang->line('Successfully deleted');}
        else {
            $response['success'] = false;
            $response['messages'] = $this->lang->line('Error in the database while deleting the information');}
        }
```

In this case, District can be found in more than one tables.

```
//---> Validate if the district is used in table Beekeeper, Association or Apiary
public function checkIntegrity($id)
{

    $num_rows = 0;

    $sql = "SELECT * FROM beekeeper WHERE district_id = ?";
    $query = $this->db->query($sql, array($id));
    $num_rows = $num_rows + $query->num_rows();

    $sql = "SELECT * FROM association WHERE district_id = ?";
    $query = $this->db->query($sql, array($id));
    $num_rows = $num_rows + $query->num_rows();

    $sql = "SELECT * FROM apiary WHERE district_id = ?";
    $query = $this->db->query($sql, array($id));
    $num_rows = $num_rows + $query->num_rows();

    return $num_rows;

}
```

Form validation parameter

Here you have the information about the rules of the library form_validation

Rule	Parameter	Description	Example
required	No	Returns FALSE if the form element is empty.	
matches	Yes	Returns FALSE if the form element does not match the one in the parameter.	matches[form_item]
regex_match	Yes	Returns FALSE if the form element does not match the regular expression.	regex_match[/regex/]
differs	Yes	Returns FALSE if the form element does not differ from the one in the parameter.	differs[form_item]
is_unique	Yes	Returns FALSE if the form element is not unique to the table and field name in the parameter. Note: This rule requires Query Builder to be enabled in order to work.	is_unique[table.field]
min_length	Yes	Returns FALSE if the form element is shorter than the parameter value.	min_length[3]
max_length	Yes	Returns FALSE if the form element is longer than the parameter value.	max_length[12]
exact_length	Yes	Returns FALSE if the form element is not exactly the parameter value.	exact_length[8]
greater_than	Yes	Returns FALSE if the form element is less than or equal to the parameter value or not numeric.	greater_than[8]
greater_than_equal_to	Yes	Returns FALSE if the form element is less than the parameter value, or not numeric.	greater_than_equal_to[8]
less_than	Yes	Returns FALSE if the form element is greater than or equal to the parameter value or not numeric.	less_than[8]
less_than_equal_to	Yes	Returns FALSE if the form element is greater than the parameter value, or not numeric.	less_than_equal_to[8]
in_list	Yes	Returns FALSE if the form element is not within a predetermined list.	in_list[red,blue,green]
alpha	No	Returns FALSE if the form element contains anything other than alphabetical characters.	
alpha_numeric	No	Returns FALSE if the form element contains anything other than alpha-numeric characters.	
alpha_numeric_spaces	No	Returns FALSE if the form element contains anything other than alpha-numeric characters or spaces. Should be used after trim to avoid spaces at the beginning or end.	
alpha_dash	No	Returns FALSE if the form element contains anything other than alpha-numeric characters, underscores or dashes.	
numeric	No	Returns FALSE if the form element contains anything other than numeric characters.	
integer	No	Returns FALSE if the form element contains anything other than an integer.	
decimal	No	Returns FALSE if the form element contains anything other than a decimal number.	
is_natural	No	Returns FALSE if the form element contains anything other than a natural number: 0, 1, 2, 3, etc.	
is_natural_no_zero	No	Returns FALSE if the form element contains anything other than a natural number, but not zero: 1, 2, 3, etc.	
valid_url	No	Returns FALSE if the form element does not contain a valid URL.	
valid_email	No	Returns FALSE if the form element does not contain a valid email address.	
valid_emails	No	Returns FALSE if any value provided in a comma separated list is not a valid email.	
valid_ip	Yes	Returns FALSE if the supplied IP address is not valid. Accepts an optional parameter of 'ipv4' or 'ipv6' to specify an IP format.	