**Codeighniter**

Helper

Cara mendapatkan nilai dari $post biar gampang langsung di simpan ke database:

$this->post\_model->update(

elements(array('id', 'title', 'content'), $\_POST)

);

Penjelasan:

Id, title, content : nama input yang dari form

$\_post : array yang berisi input id, title, content.

Kalau biasanya untuk mendapatkan nilai dari form, kita perlu mendapatkan dengan cara yg sedikit lebih Panjang.

$id = $this->input->post(‘id’);

Mendapatkan isi dari list directory:

$this->load->helper('directory');

        $map = directory\_map(\_\_DIR\_\_);

        print\_r ($map);

outputnya adalah array. Contoh:

Array ( [0] => Admin\_C.php [1] => index.html [2] => Latihan\_c.php [3] => Product\_C.php [4] => Sale\_c.php [5] => Setting\_Admin\_C.php [6] => Website\_c.php )

Cara mencreate 1 file dan langsung download

$this->load->helper('download');

        $data = 'Here is some text!';

        $name = 'mytext.txt';

        force\_download($name, $data);

itu nanti akan mendowload 1 file yang di create di code itu juga denga nisi here is some text dan nama filenya adalah mytext.txt

membaca isi dari sebuah file

public function file()

    {

        $this->load->helper('file');

        $string = read\_file(\_\_DIR\_\_."\Latihan\_c.php");

        echo $string;

    }

Fungsi read\_file Sudah depreceated, sekarng pakai file\_get\_contents() saja.

Menghapus semua file dalam sebuah folder, hanya akan menghapus file.

delete\_files('./path/to/directory/');

jika ingin menghapus folder juga di dalam sebuah directory (menghapus folder juga):

delete\_files('./path/to/directory/', TRUE);

mengetahui permission dari sebuah file dalam huruf

echo symbolic\_permissions(fileperms('./index.php')); // -rw-r--r—

mengetahui permission dari sebuah file dalam angka

echo octal\_permissions(fileperms('./index.php')); // 644

form\_open()

format:

form\_open([$action = ''[, $attributes = ''[, $hidden = array()]]])[¶](file:///C:\xampp\htdocs\ris\user_guide\helpers\form_helper.html#form_open)

contoh:

echo form\_open('email/send');\

ini sama dengan html form berikut:

<form method="post" accept-charset="utf-8" action="http://example.com/index.php/email/send">

Jika kita ingin menambah attribute:

$attributes = array('class' => 'email', 'id' => 'myform');

echo form\_open('email/send', $attributes);

menambah parameter element hidden di url sebagai parameter yang dimana akan mengirim semua element yang hidden di dalam element tersebut.

$hidden = array('username' => 'Joe', 'member\_id' => '234');

echo form\_open('email/send', '', $hidden);

contoh:

<form method="post" accept-charset="utf-8" action="http://example.com/index.php/email/send">

<input type="hidden" name="username" value="Joe" />

<input type="hidden" name="member\_id" value="234" />

form untuk bisa upload file.

Untuk bisa mengupload file kita menggunakan form\_open\_multipart.

Format :

form\_open\_multipart([$action = ''[, $attributes = array()[, $hidden = array()]]])

contoh:

di form:

<input type="file" name="berkas" />

Di file untuk save:

$config['upload\_path']          = './uploads/';

    $config['allowed\_types']        = 'gif|jpg|png';

    $config['max\_size']             = 100;

    $config['max\_width']            = 1024;

    $config['max\_height']           = 768;

    $this->load->library('upload', $config);

    $this->upload->do\_upload('image');

Sedangkan untuk menyimpan nama file yang di upload ke database, kita bisa gunakan seperti berikut:

$image = $this->upload->data('file\_name');

ini kita buat dibawah code setelah proses upload. Jadi urutannya seperti berikut:

$updata = $this->upload->do\_upload('image')

$image = $this->upload->data('file\_name');

Membuat field seperti ini:

<input type="text" name="username" value="johndoe" id="username" maxlength="100" size="50" style="width:50%" />

Kita bisa menggunakan bantuan helper CI dengan code berikut langsung dari php

$data = array(

'name' => 'username',

'id' => 'username',

'value' => 'johndoe',

'maxlength' => '100',

'size' => '50',

'style' => 'width:50%'

);

echo form\_input($data);

kita juga bisa menambahkan fungsi javascript, seperti onclick:

$js = 'onClick="some\_function()"';

echo form\_input('username', 'johndoe', $js);

atau dengan cara berikut:

$js = array('onClick' => 'some\_function();');

echo form\_input('username', 'johndoe', $js);

form input dengan type password:

form\_password([$data = ''[, $value = ''[, $extra = '']]])[¶](file:///C:\xampp\htdocs\ris\user_guide\helpers\form_helper.html#form_password)

**text-area**

form\_textarea([$data = ''[, $value = ''[, $extra = '']]])[¶](file:///C:\xampp\htdocs\ris\user_guide\helpers\form_helper.html#form_textarea)

form\_dropdown([$name = ''[, $options = array()[, $selected = array()[, $extra = '']]]])[¶](file:///C:\xampp\htdocs\ris\user_guide\helpers\form_helper.html#form_dropdown)

contoh:

$options = array(

'small' => 'Small Shirt',

'med' => 'Medium Shirt',

'large' => 'Large Shirt',

'xlarge' => 'Extra Large Shirt',

);

$shirts\_on\_sale = array('small', 'large');

echo form\_dropdown('shirts', $options, 'large');

output:

<select name="shirts">

<option value="small">Small Shirt</option>

<option value="med">Medium Shirt</option>

<option value="large" selected="selected">Large Shirt</option>

<option value="xlarge">Extra Large Shirt</option>

</select>

$js = 'id="shirts" onChange="some\_function();"';

echo form\_dropdown('shirts', $options, 'large', $js);

form\_multiselect([$name = ''[, $options = array()[, $selected = array()[, $extra = '']]]])[¶](file:///C:\xampp\htdocs\ris\user_guide\helpers\form_helper.html#form_multiselect)

untuk semua type field input langsung check di documentation Codeighniter.

Melihat id yang baru di insert:

**$this->db->insert\_id()**

Contoh:

$res = $this->db->insert($table\_name, $data);

$inserted\_id = $this->db->insert\_id();

var\_dump($inserted\_id);

Melihat jumlah id yang kena effect baik insert, update, dll.

**$this->db->affected\_rows()**

Melihat query terakhir:

**$str = $this->db->last\_query()**

**Echo $str**

**Note:**

Disabling the **save\_queries** setting in your database configuration will render this function useless.

Menghitung jumlah data/baris dari sebuah table

**$this->db->count\_all()**

Contoh:

echo $this->db->count\_all('my\_table');

membuat query sendiri

**$this->db->insert\_string()**

Contoh:  
$data = array('name' => $name, 'email' => $email, 'url' => $url);

$str = $this->db->insert\_string('table\_name', $data);

Untuk update:

**$this->db->update\_string()**

Contoh:

$data = array('name' => $name, 'email' => $email, 'url' => $url);

$where = "author\_id = 1 AND status = 'active'";

$str = $this->db->update\_string('table\_name', $data, $where);

**Query builder**

**$this->db->get\_compiled\_select()**

Code di atas tidak menjalankan query, itu hanya membentuk query string. Dengan begitu kita bisa melihat query yang di execute.

Contoh:

$query = **$this->db->get\_compiled\_select()**

**echo $query;**

query akan tergabung seperti berikut:

$this->db->limit(10,20)->get\_compiled\_select('mytable', FALSE);

Kemudian kita buat lagi query:

$this->db->select('title, content, date')->get\_compiled\_select();

Maka query pertama akan digabung dengan query ke 2, yaitu limit akan lengket di query ke 2.

Maka hasilnya adalah:

SELECT title, content, date FROM mytable LIMIT 20, 10

Untuk menghindari itu, kita bisa reset querynya.

Yaitu dengan menambah parameter ke 2 Di query pertama:

. Seperti berikut ini:

$this->db->limit(10,20)->get\_compiled\_select('mytable', FALSE);\

Kita juga bisa bentuk query kita sendiri.

**$this->db->select()**

**Contoh:**

$this->db->select('title, content, date');

$query = $this->db->get('mytable');

Mencari value tertinggi. Misalnya untuk id. biasanya ini kita untuk id terakhir di table.

**$this->db->select\_max()**

**Contoh:**

SELECT MAX(field)

**Yang paling kecil**

**$this->db->select\_min()**

Mencari rata-rata:

**$this->db->select\_avg()**

Menjumlah:

**$this->db->select\_sum()**

Join:

**$this->db->join()**

**Contoh:**

$this->db->select('\*');

$this->db->from('blogs');

$this->db->join('comments', 'comments.id = blogs.id');

$query = $this->db->get();

Left join

$this->db->join('comments', 'comments.id = blogs.id', 'left');

Mengambil data tertentu:

**$this->db->where()**

**Contoh:**

$this->db->where('name', $name);

$this->db->where('title', $title);

$this->db->where('status', $status);

Kita juga bisa custome nilai dari where itu:

Misalnya where != .

this->db->where('name !=', $name);

atau dengen query kita sendiri:

$where = "name='Joe' AND status='boss' OR status='active'";

$this->db->where($where);

**Or\_where**

**$this->db->or\_where()**

**Contoh:**

$this->db->where('name !=', $name);

$this->db->or\_where('id >', $id); //

Output:

Produces: WHERE name != 'Joe' OR id > 50

Contoh lain:

public function edit\_ticket($table\_name, $id)

  {

    $this->db->select("$table\_name.\*");

    $this->db->select('client.full\_name as fname, client.id');

    $this->db->from($table\_name);

    $this->db->join('client', "client.id = $table\_name.client", 'left');

    $this->db->where("$table\_name.id", $id);

    return $this->db->get()->row();

  }

Where in

**$this->db->where\_in()**

**Contoh:**

$names = array('Frank', 'Todd', 'James');

$this->db->where\_in('username', $names);

**$this->db->or\_where\_in()**

$names = array('Frank', 'Todd', 'James');

$this->db->or\_where\_in('username', $names);

// Produces: OR username IN ('Frank', 'Todd', 'James')

**$this->db->where\_not\_in()**

**$this->db->or\_where\_not\_in()**

**$this->db->like()**

**Simple key/value method:**

$this->db->like('title', 'match');

// Produces: WHERE `title` LIKE '%match%' ESCAPE '!'

If you use multiple method calls they will be chained together with AND between them:

$this->db->like('title', 'match');

$this->db->like('body', 'match');

// WHERE `title` LIKE '%match%' ESCAPE '!' AND `body` LIKE '%match% ESCAPE '!'

If you want to control where the wildcard (%) is placed, you can use an optional third argument. Your options are ‘before’, ‘after’, ‘none’ and ‘both’ (which is the default).

$this->db->like('title', 'match', 'before'); // Produces: WHERE `title` LIKE '%match' ESCAPE '!'

$this->db->like('title', 'match', 'after'); // Produces: WHERE `title` LIKE 'match%' ESCAPE '!'

$this->db->like('title', 'match', 'none'); // Produces: WHERE `title` LIKE 'match' ESCAPE '!'

$this->db->like('title', 'match', 'both'); // Produces: WHERE `title` LIKE '%match%' ESCAPE '!'

**$this->db->group\_by()**

**Contoh:**

$this->db->group\_by("title");

Bisa juga lebih dari 1

$this->db->group\_by(array("title", "date"));

Mencari nilai yang tidak duplicate

**$this->db->distinct()**

$this->db->distinct();

$this->db->get('table'); // Produces: SELECT DISTINCT \* FROM table

Order\_by

**$this->db->order\_by()**

$this->db->order\_by('title', 'DESC');

// Produces: ORDER BY `title` DESC

Multiple order

$this->db->order\_by('title', 'DESC');

$this->db->order\_by('name', 'ASC');

// Produces: ORDER BY `title` DESC, `name` ASC

Atau juga bisa langsung dengan:

$this->db->order\_by('title DESC, name ASC');

// Produces: ORDER BY `title` DESC, `name` ASC

Membatasi jumlah yang deselect

**$this->db->limit()**

$this->db->limit(10);

Menghitung jumlah yg deselect

$**this->db->count\_all\_results()**

**Contoh:**

echo $this->db->count\_all\_results('my\_table'); // Produces an integer, like 25

jika kita ingin select table tertentu, maka untuk countall nya adalah:

**$this->db->count\_all()**

**Contoh:**

echo $this->db->count\_all('my\_table'); // Produces an integer, like 25

insert:

insert single data

**$this->db->insert()**

$data = array(

'title' => 'My title',

'name' => 'My Name',

'date' => 'My date'

);

$this->db->insert('mytable', $data);

// Produces: INSERT INTO mytable (title, name, date) VALUES ('My title', 'My name', 'My date')

Insert multiple data

**$this->db->insert\_batch()**

**Contoh:**

$data = array(

array(

'title' => 'My title',

'name' => 'My Name',

'date' => 'My date'

),

array(

'title' => 'Another title',

'name' => 'Another Name',

'date' => 'Another date'

)

);

$this->db->insert\_batch('mytable', $data);

// Produces: INSERT INTO mytable (title, name, date) VALUES ('My title', 'My name', 'My date'), ('Another title', 'Another name', 'Another date'

[**Method Chaining**](file:///C:\xampp\htdocs\ris\user_guide\database\query_builder.html#id10)[**¶**](file:///C:\xampp\htdocs\ris\user_guide\database\query_builder.html#method-chaining)

Method chaining allows you to simplify your syntax by connecting multiple functions. Consider this example:

$query = $this->db->select('title')

->where('id', $id)

->limit(10, 20)

->get('mytable');

Caching

### An example of caching[¶](file:///C:\xampp\htdocs\ris\user_guide\database\query_builder.html#an-example-of-caching)

Here’s a usage example:

$this->db->start\_cache();

$this->db->select('field1');

$this->db->stop\_cache();

$this->db->get('tablename');

//Generates: SELECT `field1` FROM (`tablename`)

$this->db->select('field2');

$this->db->get('tablename');

//Generates: SELECT `field1`, `field2` FROM (`tablename`)

$this->db->flush\_cache();

$this->db->select('field2');

$this->db->get('tablename');

//Generates: SELECT `field2` FROM (`tablename`)

Caching

Contoh:

// Turn caching on

$this->db->cache\_on();

$query = $this->db->query("SELECT \* FROM mytable");

// Turn caching off for this one query

$this->db->cache\_off();

$query = $this->db->query("SELECT \* FROM members WHERE member\_id = '$current\_user'");

// Turn caching back on

$this->db->cache\_on();

$query = $this->db->query("SELECT \* FROM another\_table");\

Code di atas akan menghasil/mengcreate 1 folder dan file dengan nama:

Nama\_controller+nama\_function dan ada filenya di dalam

Note: yang bisa dicaching hanya query **select**

# Transactions

Transaction memiliki fungsi untuk membuat blok untuk multiple query. Maksudnya adalah sebuah query tidak akan di eksekusi sebelum di script trans\_complete. Maksudnya bagaimana? Memang tanpa transactions, querynya langsung dijalankan?

Jawab:

Pertama untuk konsep query adalah:

Query bejalan procedural, dimana akan menjalankan baris mulai dari atas. Contoh jika kita ada query, sperti berikut:

$link = mysqli\_connect("localhost", "root", "", "demo");

$sql = "INSERT INTO persons (first\_name, last\_name, email) VALUES ('Peter', 'Parker', 'peterparker@mail.com')";

mysqli\_query($link, $sql)

ketika, pembacaan mysqli\_query ($link, Sql) sudah dibaca, itu akan di eksekusi.

Untuk single query itu tidak menjadi masalah. Tapi jika methode seperti itu (saat baca langsung eksekusi) itu akan menjadi masalah ketika kita melakukan lebih dari 1 query untuk menginsert/update ke database. kenapa? Karena jika terjadi error, saat di query ke 2, data yang di insert akan menjadi miss karena ada data yang masuk ke database dan ada yang tidak. Tapi lebih baik itu, tidak masuk sama sekali, atau masuk semua.

Nah untuk mengatasi seperti itu kita pakai methode transactions.

Contoh :

$this**->**db**->**trans\_start();

$this**->**db**->**query('AN SQL QUERY...');

$this**->**db**->**query('ANOTHER QUERY...');

$this**->**db**->**query('AND YET ANOTHER QUERY...');

$this**->**db**->**trans\_complete();

**Bencmark**

**Contoh:**

$this->benchmark->mark('code\_start');

        $this->db->get('product');

        $this->benchmark->mark('code\_end');

        echo $this->benchmark->elapsed\_time('code\_start', 'code\_end');

**Output:**

0.0028

Menampilkan lama waktu process di view:

<?php echo $this->benchmark->elapsed\_time();?>

Atau bisa langsung:

{elapsed\_time}

Menampilkan pemakaian limit memory

<?php echo $this->benchmark->memory\_usage();?> # ini pemakain memory untuk semua aplikasi.

Atau bisa langsung:

{memory\_usage}

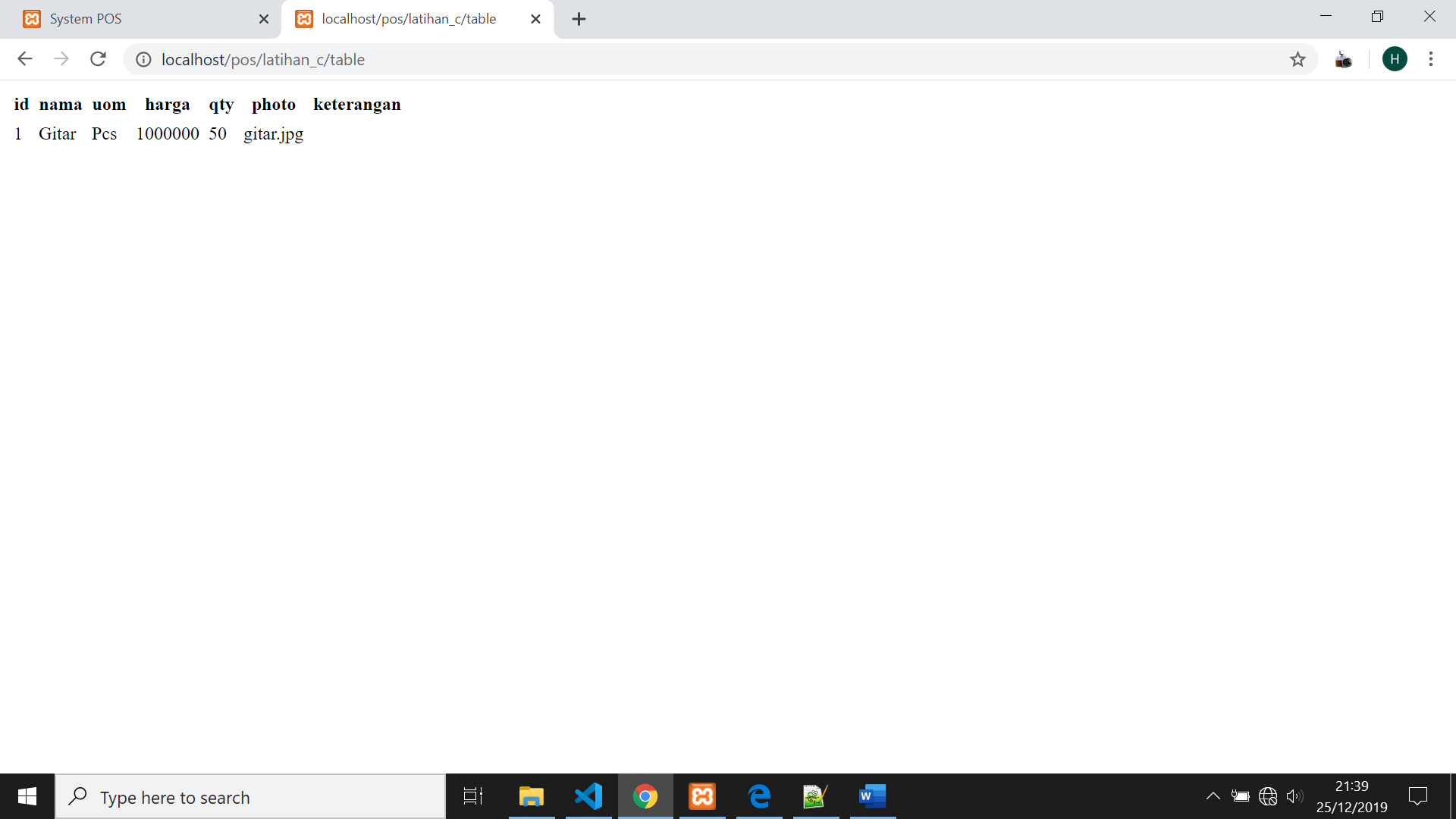
Menampilkan table langsung dari result query:

$this->load->library('table');

        $query = $this->db->query('SELECT \* FROM product');

        echo $this->table->generate($query);

output:



**Form Validation**

Untuk membuat form validation dengan codeighneter, kita membutuhkan 2 library, yaitu: **form** dan **form\_validation** dari codeighneter untuk di load.

Maka langkah-langkahnya adalah :

1. Load library yang di butuhkan : **form** dan **form\_validation**

**2. P**anggil function set\_rules dari form\_validation. $this**->**form\_validation**->**set\_rules

(). Di sini ada 3 parameter, Yaitu : nama field, pesan eror, dan apa yang menjadi validasinya (required, unik, dll). Contoh: $this**->**form\_validation**->**set\_rules('username', 'Username is required', 'required');

3. Terakhir jalankan fungsi run yang ada di form validation. $this**->**form\_validation**->**run(). hasil return dari fungsi run ini adalah true atau false. Jika untuk mengecek ada error untuk memberi kondisi, kemana view akan di redirect bisa di cek dari situ.

Contoh:

**if** ($this**->**form\_validation**->**run() **==** **FALSE**)

{

$this**->**load**->**view('myform');

}

**else**

{

$this**->**load**->**view('formsuccess');

}

**Membuat Alert setelah tekan button SUBMIT (save, edit, delete)**

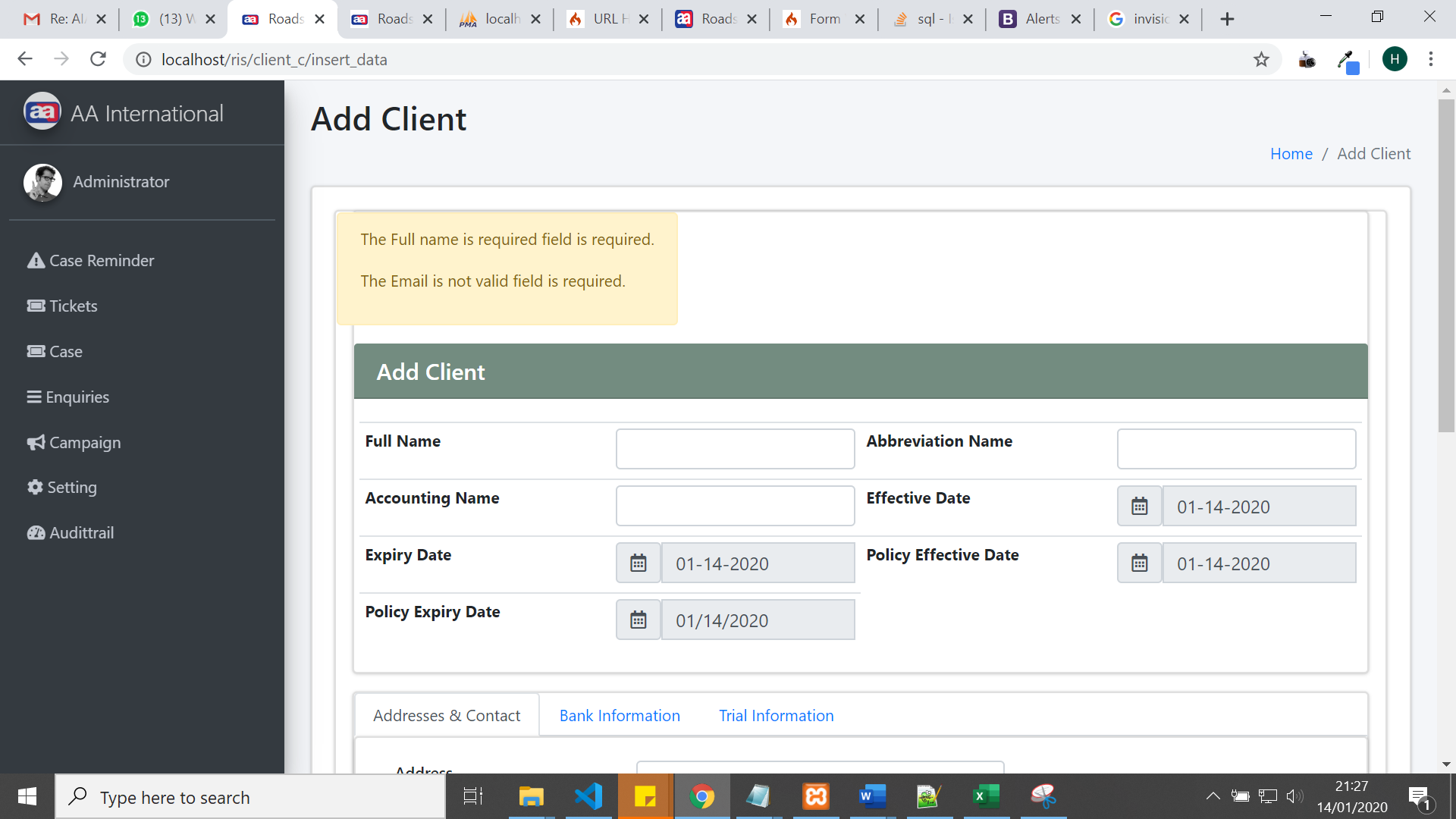
Note: kita membutuhkan 1 element, di sini kita akan memakai class dari bootstrap (alert alert-info)

Dalam pembuatan warning ini, dipastikan kita sudah membuat form validation menggunakan library codeighneter. Karena kita akan mengecek errornya dengan function validation\_errors(). Dan fungsi itu adalah dating dari library form\_validation.

Alert Error:

Flow yang akan di buat:

1. Element Alert hanya akan tampil jika validation\_error ada nilainya. Perlu kita tahu, jika tidak ada error, maka value dari function validation\_errors() adalah string kosong (“”). Kita bisa cek itu dengan var\_dump.
2. Jika bernilai string kosong, maka element error tidak akan tampil.



Sedangkan untuk pembuatan alert success, kita bisa cek dari hasil dari function validation\_errors nya maupun dari hasil insert dari funsgi insert:

$this->client\_m->insert\_data('client', $data);

**Membuat Pesan setelah tekan tombol submit**

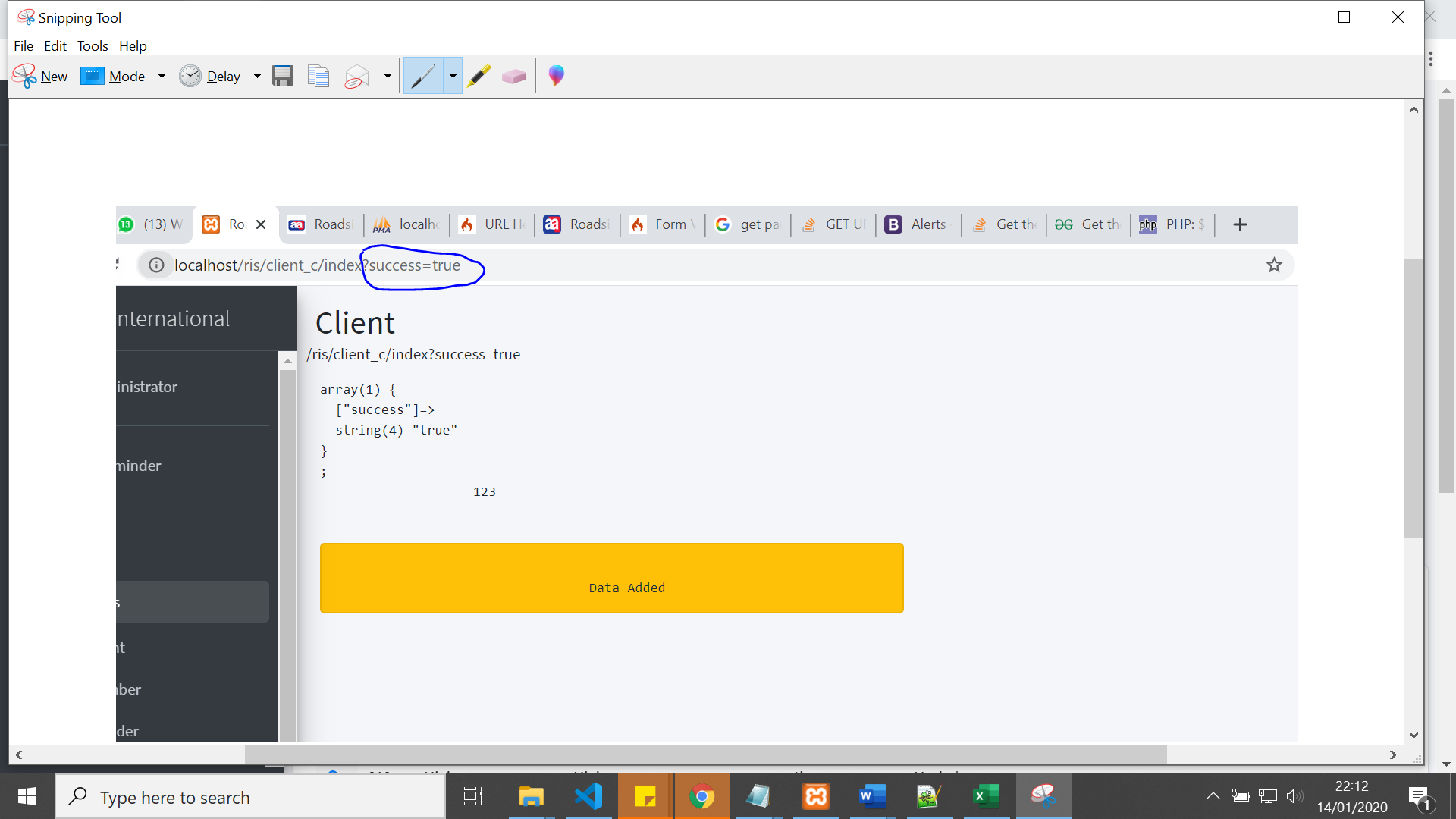
Untuk membuat pesan success setelah tekan tombol submit.

Flow:

1. Ketika kita, tekan button submit misalnya pada saat simpan data, maka akan redirect ke list view. Dan di listviewnya akan muncul pesan data “berhasil di tambah”.

2. Pesan itu akan muncul ketika di parameter url ada success=true. Ini nanti kita tambah di redirect setelah submit.

Contoh:



Cara membuatnya:

Di form:

Ada process form dan ada button submit.

Kemudian di controller:

        $insert = $this->client\_m->insert\_data('client', $data);

        redirect(base\_url() . 'client\_c/index?success="true"');

redirect di atas akan meredirect halaman ke listview client dan di urlnya di sertakan dengan key success=”true”.  
seperti kita perhatikan di atas, untuk menambahkan parameter itu harus di awali dengan tanda tanya (?). kalau tidak itu nanti tidak akan terdeteksi ketika echo $\_GET[‘success’].

Kemudian di listview clientnya

<div>

                    <?php echo $\_SERVER['REQUEST\_URI'];

                    ?>;

                    <?php if (isset($\_GET['success'])) {

                        echo 123;

                    ?>

                        <div>

                            <div class="alert alert-warning" role="alert">

                                <?php echo 'Data Added'; ?>

                            </div>

                        </div>

                    <?php } ?>

                </div>

**Session**

a. Initialization

$this->load->library('session');

b. Set session

Datanya kita buat array

$new\_data = array(

       'info'=> 'create\_new',

            );

$this->session->set\_userdata($new\_data);

c. cara memanggil

ada 3 cara memanggilnya:

* $\_**SESS**ION['item']
* $this**->sess**ion**->**item
* $this**->sess**ion**->**userdata('item');

Item daam hal ini adalah nama key di array.

Maka dari contoh di atas, kita akan panggil dengan cara:

$this->session->info

Untuk melihat nya kita bisa echo $this->session->info.

Info adalah key saat di set sessionnya.

Setelah kita membuat pesan success dengan class **toast** dari bootstrap, kita ingin pesan tidak muncul ketika kita reload pagenya. Maka untuk membuat seperti itu kita memakai session untuk mencek itu apakah klik tombol submit atau hanya sekedar refresh/ reload halaman saja. Jika itu hanya sekedar halaman kita tidak perlu menampilkan pesan tersebut. Maka kita nanti hapus session di fungsi index di dalam controllernya, setelah semua view nya di load.

Cara membuat join di datatables query:

var $table = 'provider'; //nama tabel dari database

    var $column\_order = array(null, 'status', 'provider\_code', 'full\_name','accounting\_name', 'c\_name'); //field yang ada di table user

    var $column\_search = array('status', 'provider\_code', 'full\_name'); //field yang diizin untuk pencarian

    var $order = array('full\_name' => 'asc'); // default order

public function get\_detail($table\_name, $id)

    {

        $this->db->select("$table\_name.\*");

        $this->db->select('client\_status.name as cs\_name');

        $this->db->select('provider\_type.name as pt\_name');

        $this->db->from($table\_name);

        $this->db->join('client\_status',"client\_status.id = $table\_name.status", 'left');

        $this->db->join('provider\_type',"provider\_type.id = $table\_name.type", 'left');

        $this->db->where("$table\_name.id", $id);

        return $this->db->get()->row();

    }

catatan: jika menampung 2 nilai dalam arti di tampilkan charter tapi yang di simpan adalah id nya maka bagusnya di buat selection box aja, biar bisa menampung itu. Contohnya untuk country di provider.

Error experience:

Unable to locate the specified class: Session.php

Itu karena kita salah extend parent,

Contoh:

Kita buat model tapi extends ke controller.

Contoh:

class Chat\_model extends CI\_Controller

harusnya:

class Chat\_model extends CI\_Model

Didalam controller tidak boleh, function yang kita buat tidak boleh sama dengan nama classnya.

Karena itu akan membuat error.

Contoh yang salah:

<?php

class Home extends CI\_Controller{

    public function index()

    {

        return $this->home();

    }

    public function home()

    {

        $this->load->view('template/header');

        $this->load->view('home');

        $this->load->view('template/footer');

    }

}

Seperti yang kita lihat nama class nya adalah Home, di dalam class itu ada function home,

Maka itu akan menimbulkan error.

Autocomplete dengan prosedur MVC yang benar. Prosedur yang benar maksudnya adalah dimana dari view akan terkoneksi ke controller dulu dan controller akses ke model/database.

View->Controller->model. dimana sebelumnya dari view saya langsung akses ke model/database.

Contoh salah yang biasa saya lakukan adalah:

<select id="s\_category" class="form-control">

  <option>Please Select</option>

  <?php

   $datas = $this->category\_m->get\_all\_data('category');

           foreach ($datas as $data) {

           ?>

          echo "<option value="<?php echo $data->id; ?>"><?php echo $data->name; ?></option>";

          <?php

           }

           ?>

</select>

Gambar 1

Coding di atas memang menghasilkan hasil yang benar, tapi kalau kita lihat code di atas, itu menyalahi aturan MVC. Dimana dari view langsung loncat ke model/database. Seharusnya dari controller dulu baru ke model. kemudian return dari model di kembalikan ke controller, dan hasilnya di echo di controller.

Cara yang benar:

View tag select (element html)

<select class="form-control" name="vehicle\_brand" id="vehicle\_brand">

     <option>Please Select</option>

</select>

Di javascriptnya (kita akan panggil dengan ajax):

$.ajax({

        url: "<?php echo base\_url() . 'vehicle\_brand\_c/select\_complete' ?>",

        method: "POST",

        data: [],

        dataType: 'html',

      })

      .done(function(data) {

        try {

          $('#vehicle\_brand').html(data);

        } catch (err) {

          alert(err.message);

        }

      });

Dari url source, menuju ke controller:

url: "<?php echo base\_url() . 'vehicle\_brand\_c/select\_complete' ?>",

kemudian kita perhatikan di controllernya:

public function select\_complete()

    {

        $res = $this->vehicle\_brand\_m->select\_complete('vehicle\_brand');

        if (count($res) > 0) {

            foreach ($res as $row) {

                $arr\_res[] = $row->name;

                echo "<option value='$row->id'>$row->name</option>";

            }

        }

    }

Kalau kita lihat, outpurnya adalah tag element option html:

echo "<option value='$row->id'>$row->name</option>";

kemudian kita perhatikan function select\_complete di model vehicle\_brand\_m:

public function select\_complete($table\_Name)

    {

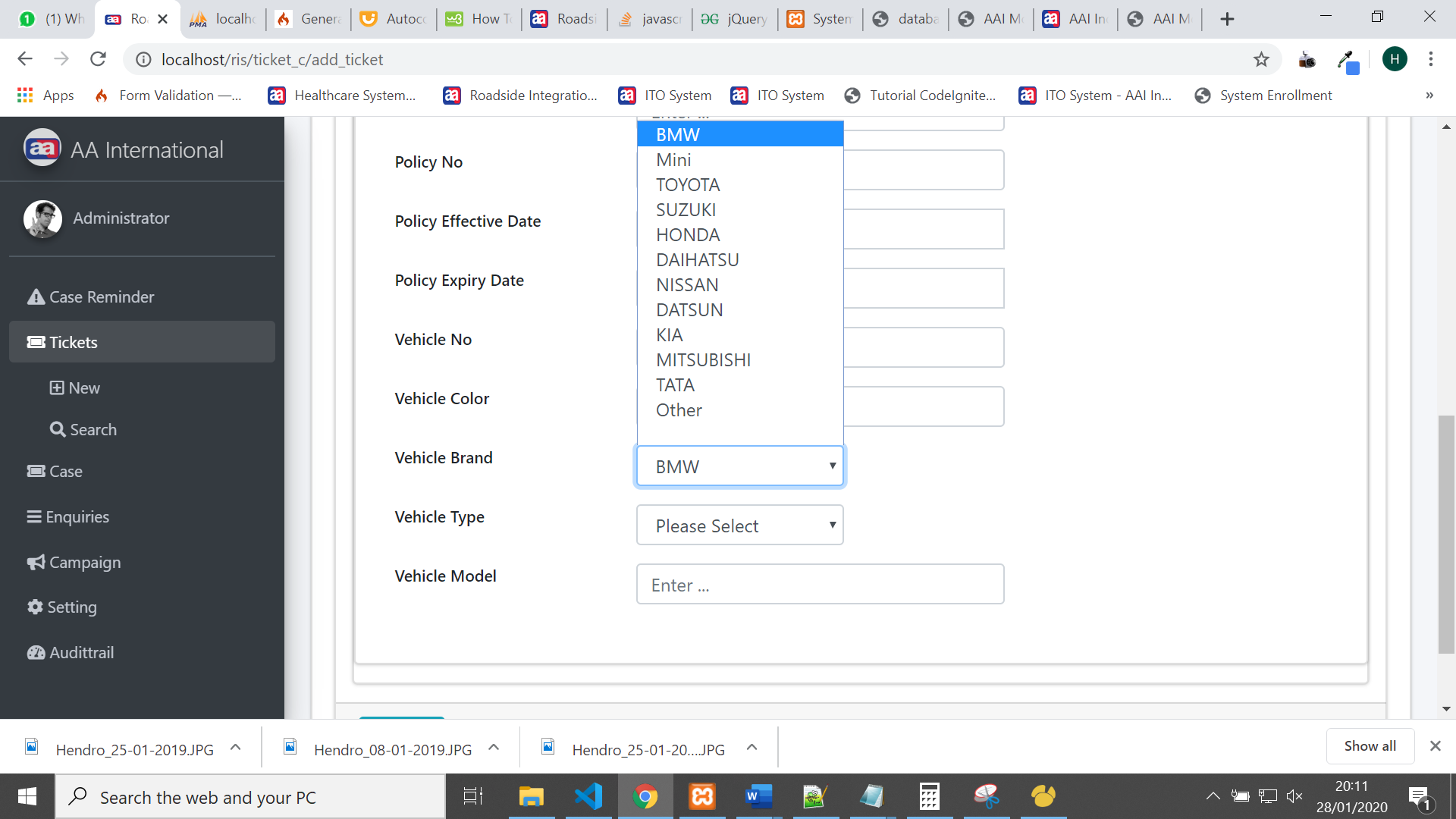
        $this->db->select('id,name');

        return $this->db->get($table\_Name)->result();

    }

Hasilnya sama, yaitu selectionnya akan tetap menghasilkan output yang sama di dalam field selection box nya.

Hasil dari antara ke duanya:



Cara clone/menggabungkan template dari luar (yang sudah jadi milik orang lain) kedalam codeighniter. Contoh , ketika kita ingin menggabungkan dengan template login dengan codeighniter kita.

Cara nya:

1. Download templatenya dari internet
2. Copy paste templatenya ke dalam asset (nama\_module/assets/new\_folder/)
3. Copy file utamanya(tampilan loginnya) ke dalam folder application/views/new\_folder
4. Pisahkan antara header dan footer dari halam utama tersebut. Header buat ke dalam file baru dengan nama header.php dan footer ke dalam footer.php. Tujuan membuat kedalam extention php supaya gampang untuk nglink ke css maupun jsnya.
5. Tambahkan sebuah controllernya
6. Di file routes setting default routsnya adalah controller login yang baru kita buat.

Cara buat where langsung dari query:

Contoh:

$where = "`case\_activity\_log`.`create\_date` IN (

$this->db->where($where);

**Routing**

Routing adalah proses mengatur untuk mengarahkan sebuah link untuk halaman/page website kita ke satu halaman tertentu. Di dalam codeighniter pengaturan routes itu ada di dalam folder: application/config/routes.php .

By default isi dari routes.php codeighniter versi 3

$route['default\_controller'] = 'welcome';

$route['404\_override'] = '';

$route['translate\_uri\_dashes'] = FALSE;

Seperti yang kita lihat di atas, di line pertama itu adalah default\_controller, default\_controller berguna sebagai default halaman utama website kita Ketika kita menuliskan domain saja di url.

Contoh: nama website/ domain kita adalah http://devwebtech.com, maka yang di tampilin adalah controller welcome/index.

Penjelasan dari routes di atas:

$route['default\_controller'] = 'welcome';

default\_controller : key dari codeighniter sendiri untuk default route

‘welcome’ : nama controller kita yang dimana nanti default urlnya yang di pakai adalah Welcome.php dengan default fungsi adalah index. Jadi lengkapnya adalah ‘welcome/index’.

*Note: sebuah pemanggilan link controller tanpa menyertakan fungsinya maka itu akan mengambil fungsi default index*.

Contoh route yang lebih bagus:

$route['case/(:num)'] = 'case\_c/detail/$1';

Penjelasan:

Jika URI, kita menggunakan adalah : case/{angka}, maka itu diarahin ke: case\_c/detail/$1.

Berarti misalnya, url normal kita adalah http://devwebtech.com/ case\_c/detail/$1, maka kita dengan route seperti diatas, kita dapat menulis lebih simple lagi dengan : <http://devwebtech.com/case/1>.

Catatan: num adalah menyatakan number di php.

**Helper form**

Untuk membuat sebuah helper form di view supaya bisa berjalan:

Load dulu helper form (boleh dari configuration autoload atau bisa juga di controller di class).

Kemudian di viewnya:

Pembuka:

<?php echo form\_open(base\_url() . 'Casehpm\_c/update') ?>

Penutup:

<?php echo form\_close(); ?>

Pastikan jika setelah pembuka dan penutup ada 1 blok tag <div> </div>

Jadi formatnya:

<?php echo form\_open(base\_url() . 'Casehpm\_c/update') ?>

    <div>

Isi field

</div>

    <?php echo form\_close(); ?>

kalau ngga begitu kemungkinan ada error.

**Cara Membuat Login, Register, Logout**

1. Login

- Buat tabel user

- Buat form (beserta validasinya)

- Ketika button login di klik, akan di cek tabel user apakah benar ada datanya, kalau benar arahkan ke halaman sesuai user level yg login. Tapi kalau salah, makan akan tetap di halaman login (redirect dari function).

2. Register

Ketika klik button daftar/ register, maka akan diarahkan ke halaman register. Kemudian setelah form register dan di submit oleh user, maka akan diarahkan kehalaman login.

3. Logout

Destroy session dan diredirect ke halaman login.