

## MODUL X-FRAMEWORK CI (BAGIAN 2)

### 10.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami cara kerja menggunakan library yang ada di framework.
2. Mampu membuat aplikasi memanfaatkan library yang ada di framework.

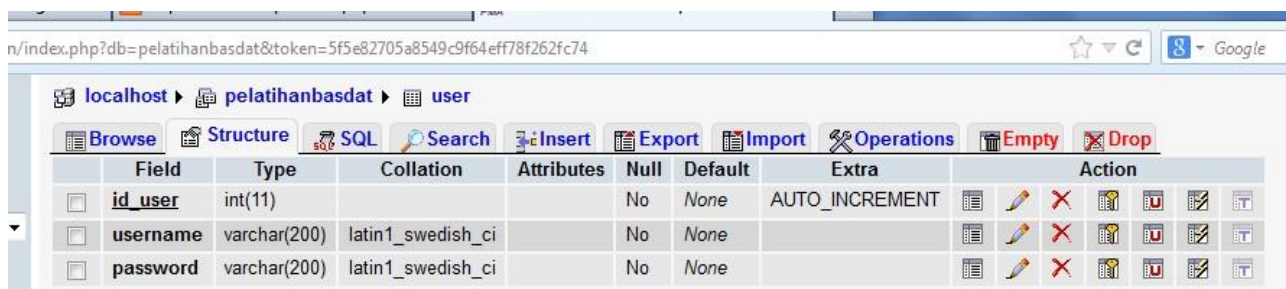
### 10.2 Implementasi Session CodeIgniter di Proses Autentikasi Akun

*Session* merupakan suatu cara merekam dan memantau aktivitas *user* dalam menggunakan aplikasi *web* yang kita bangun. Dengan *session* seseorang dapat dikenal identitasnya apakah orang tersebut mempunyai hak akses terhadap aplikasi *web* yang digunakannya, mencatat setiap *check point* yang dilalui oleh *user* (misalnya : keranjang belanja), atau menyimpan data hasil interaksi *user* dengan aplikasi *web* sebelum disimpan di dalam *storage*.

Dalam sesi kali ini, kita akan mencoba membuat sistem autentikasi sederhana. Kronologinya adalah *user* akan memasukkan akunnya, kemudian memeriksa apakah di *database* terdapat akun tersebut, jika berhasil diarahkan ke halaman sukses, jika gagal dialihkan kembali ke halaman *login*. Kemudian di setiap *function* akan diperiksa apakah *user* tersebut berhak mengakses *function* tersebut tanpa *login* atau harus *login* terlebih dahulu.

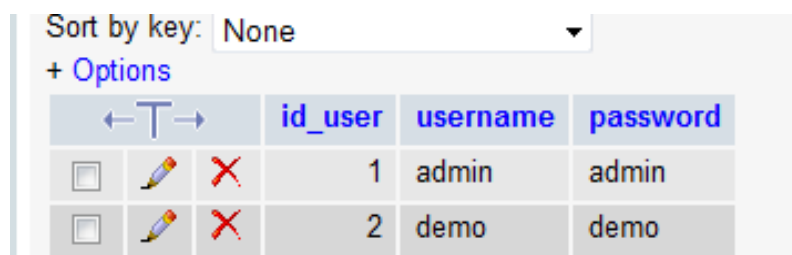
#### 10.2.1 Menyiapkan Tabel

Kita memerlukan sebuah tabel yang bernama **user**. Tabel ini mempunyai tiga buah *field* yaitu **id\_user**, **username**, **password**. Dengan menggunakan PHPMysqlAdmin, buatlah tabel **user** di *database pelatihanbasdat* dengan ketentuan seperti berikut:



Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> id_user	int(11)			No	None	AUTO_INCREMENT	
<input type="checkbox"/> username	varchar(200)	latin1_swedish_ci		No	None		
<input type="checkbox"/> password	varchar(200)	latin1_swedish_ci		No	None		

Kemudian setelah berhasil membuat tabel diatas, isikan data *dummy* seperti pada gambar berikut ini:



Sort by key: None		id_user	username	password
<input type="checkbox"/>		1	admin	admin
<input type="checkbox"/>		2	demo	demo

### 10.2.2 Membuat Model untuk Tabel User

*Model* yang akan kita perlukan untuk tabel **user** akan memiliki fungsi untuk memeriksa akun dan mengambil data akun tersebut. *Model* yang akan ditulis memiliki nama **user\_model**. Diletakkan di **application -> models -> account**. *Folder account* dibuat terlebih dahulu di dalam *folder models*. Berikut adalah *source code* dari **user\_model**:

```
<?php
class User_model extends
    CI_Model { function
        construct(){
            parent::construct();
        }
        // cek keberadaan user di sistem
        function check_user_account($username, $password){
            $this->db->select('*');
            $this->db->from('user');
            $this->db->where('username', $username);
            $this->db->where('password', $password);
            return $this->db->get();
        }
        // mengambil data
        user tertentu function
        get_user($id_user){
            $this->db->select('*');
            $this->db->from('user');
            $this->db->where('id_user', $id_user);
            return $this->db->get();
        }
    }
```

### 10.2.3 Membuat Controller Account

*Model user\_model* telah berhasil dibuat. Sekarang kita akan membuat *controller* yang bernama **account** untuk digunakan dalam membangun sistem autentikasi sederhana. *Controller* tersebut memerlukan *model user\_model* karena akan mengakses tabel **user**, memerlukan *helper url* dan *form*, serta memerlukan *library form\_validation*.

Sebagai langkah awal, kita akan membuat *function* yang menampilkan *form* login beserta *viewnya*. Sebelumnya, buat terlebih dahulu *folder account* di **application -> views**. *Folder* tersebut digunakan untuk menaruh *view* yang akan digunakan untuk sistem autentikasi sederhana ini. Berikut adalah *controller* untuk menampilkan *form login*:

```

<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Account extends
CI_Controller {

    function    construct(){
        parent::    construct();
        $this->load->model('account/user_model');
        $this->load->helper('url');
        $this->load->helper('form');
        $this->load->library('form_validation');
    }

    // melihat halaman
    login public function
    index(){
        $this->load->view('account/form_login');
    }

    .....

}
/* End of file welcome.php */
/* Location: ./application/controllers/welcome.php */

```

Pada *function index*, Anda bisa melihat bahwa *function* tersebut mengarah ke *view form\_login*. Berikut adalah isi dari *view form\_login*:

```

<h1>Silahkan Login</h1>
<fieldset>
<?php echo validation_errors(); ?>
<p style="color:red;"><?php echo $this->session->flashdata('notification')?></p>
<?php echo form_open('account/login')?>
    Username : <input type="text" name="username" value="<?php echo
    set_value('username')?>" /> <br /><br /> Password : <input type="password"
    name="password" value="<?php echo set_value('password')?>" /> <br
/><br />
    <input type="submit" name="masuk" value="Masuk" /> <br />
</form>
</fieldset>

```

Setelah selesai disalin, simpanlah *source code* diatas dengan nama **form\_login.php** dan taruh di **application -> views -> account**. Di dalam *source code* diatas terdapat beberapa poin penting seperti berikut:

- **validation\_error()**, akan menampilkan *error* yang dihasilkan ketika proses validasi *form*
- **\$this->session->flashdata('notification')**, akan menampilkan kesalahan ketika pengecekan keberadaan akun

- **set\_value()**, mencetak kembali nilai masukan di *form* sebelumnya  
Kemudian akses *function* **index** melalui URL berikut:

Maka akan muncul tampilan seperti berikut ini:

<http://localhost/pelatihanbasdat/index.php/account>



The screenshot shows a Mozilla Firefox browser window. The address bar displays the URL `localhost/pelatihanbasdat/index.php/account`. The page content features a heading **Silahkan Login** followed by a login form. The form contains two input fields: 'Username :' and 'Password :', each with a corresponding text box. Below these fields is a button labeled 'Masuk'. The browser's tab bar shows an active tab for the current URL and another tab titled 'Generating Query Results : CodeIgnit...'. The browser's menu bar includes 'File', 'Edit', 'View', 'History', 'Bookmarks', 'Tools', and 'Help'.

#### 10.2.4 Menambahkan Proses Login di Controller Account

Sekarang kita akan maju ke bagian terpenting dari proses autentikasi ini. Ketika Anda memasukkan *username* dan *password*, Anda akan mengirimkan kedua masukan tadi ke *function* **login()** di *controller* **account**. Masukan tersebut akan ditangkap menggunakan *library* **input** dan diproses juga menggunakan XSS Filtering. Lalu kedua data tersebut diperiksa di *database* apakah ada akun yang dicari atau tidak.

Selain itu diterapkan juga proses validasi terhadap *form login*. Di dalam penerapan validasi *form*, digunakan validasi **required** agar *user* tidak mengosongkan salah satu atau kedua dari isian di *form login*. Kemudian proses validasi akan menentukan apakah masukan *user* sudah lolos dari validasi *form* atau belum. Jika gagal maka akan dikembalikan ke halaman *form login*. Jika lolos maka keberadaan akun akan diperiksa.

Jika terdapat akun yang dimiliki, maka akun tersebut akan dimasukkan kedalam *session* kemudian diarahkan ke halaman sukses mengakses akun. Jika akun yang dimasukkan tidak ada dalam *database* maka *controller* **login** akan mengalihkan Anda ke halaman *form login* kembali.

Selengkapnya berikut adalah *source code* yang memproses *login* dan menyimpannya kedalam *session* setelah melewati proses validasi:

```

<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Account extends
CI_Controller {

. . . . .

. . . . .

. . . . .

    // memeriksa keberadaan akun
    username public function
    login(){
        $username = $this->input->post('username', 'true');
        $password = $this->input->post('password', 'true');

        $temp_account = $this->user_model->check_user_account($username,
        $password)->row();
        // check account
        $num_account = count($temp_account);

        $this->form_validation->set_rules('username', 'Username', 'required');
        $this->form_validation->set_rules('password',
        'Password', 'required'); if ($this-
        >form_validation->run() == FALSE)
        {
            $this->load->view('account/form_login');
        }
        else
        {
            if ($num_account > 0){
                // kalau ada set session
                $array_items = array(
                    'id_user' => $temp_account->id_user, 'username' =>
                    $temp_account->username,
                    'logged_in' => true
                );
                $this->session->set_userdata($array_items);
                redirect(site_url('account/view_success_page'));
            }
            else {
                // kalau ga ada diredirect lagi ke halaman login
                $this->session->set_flashdata('notification', 'Peringatan :
                Username dan Password
                tidak cocok');
                redirect(site_url('account'));
            }
        }
    }

. . . . .

. . . . .

. . . . .

}

/* End of file welcome.php */
/* Location: ./application/controllers/welcome.php */

```

Ketika akun yang dimasukkan ada di dalam tabel **user**. **Controller account** akan mengarahkan Anda ke halaman sukses. Sebenarnya disini terjadi pengarahannya ke **Controller account** itu sendiri yang targetnya adalah *function* yang menampilkan halaman sukses. Kita akan membuat *function* yang bernama **view\_success\_page()** di dalam **controller account**. Di dalam **controller** ini terdapat proses pengecekan *session*. Kemudian jika *session* dari *user* yang sedang login masih ada, maka akan ditampilkan halaman sukses.

Jadi ketika Anda *login* kemudian berhasil, Anda dapat mengakses *function* **view\_success\_page()** (URL : [http://pelatihanbasdat/index.php/account/view\\_success\\_page](http://pelatihanbasdat/index.php/account/view_success_page)) kembali sekalipun *tab* di *browser* sudah ditutup. Apabila sebelumnya belum *login*, kemudian Anda mengakses *function* **view\_success\_page()** maka Anda akan dialihkan ke halaman *form login*.

Berikut adalah *source code* dari *function* **view\_success\_page()**:

```
<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Account extends
CI_Controller {
. . . . .
. . . . .
. . . . .

    public function view_success_page(){
        $logged_in = $this->session-
            >userdata('logged_in'); if
            (!$logged_in){
                redirect(site_url('account'));
            }

        $this->load->view('account/success_page');
    }
. . . . .
. . . . .
. . . . .
```

Karena *function* **view\_success\_page()** akan menampilkan sebuah *view*, maka kita harus membuat sebuah *view* yang menampilkan sukses ketika *login*. Kita akan membuat sebuah *view* dengan nama **success\_page.php** kemudian simpan *view* tersebut di **application -> views -> account**. Berikut adalah *source code* dari *view* **success\_page**:

```
<h1>Anda berhasil login :D </h1>

<hr />
Hai, <?php echo $this->session->userdata('username');?> :D
<br />
Selamat Datang di website coba - coba...

<br /><br />
<a href="<?php echo site_url('account/logout'); ?>">keluar</a>

<br />
```

Sekarang waktunya kita menguji coba sistem autentikasi kita. Pertama masukkan username sesuai dengan akun yang ada di tabel **user** di database:



Kemudian jika berhasil, Anda akan diarahkan ke *function* **view\_success\_page()** seperti pada gambar berikut :



### 10.2.5 Menambahkan Fitur Logout di Controller Account

Pada subbab sebelumnya, ketika berhasil masuk ke halaman sukses. Terdapat sebuah menu untuk keluar dari sistem. Tapi menu tersebut belum dapat digunakan untuk melakukan proses *logout*.

Ketika Anda menyorot *link* tersebut, Anda akan melihat URL : <http://localhost/pelatihanbasdat/index.php/account/logout> yang artinya akan mengakses *function logout* di *controller account*.

Di *function* tersebut *session user* yang dicatat oleh sistem akan dihapus. Kemudian *function logout* akan mengalihkan Anda ke halaman *login* kembali. Berikut adalah *source code* dari *function logout*:

```
<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Account extends
CI_Controller {
. . . . .
. . . . .
. . . . .
// keluar dari
sistem public
function
logout(){
    $this->session->sess_destroy();
    redirect(site_url('account'));
}
. . . . .
. . . . .
}
```

Sekarang mari coba klik link **Keluar** yang ada di halaman sukses:





### 10.3 Mengenal Pagination di CodeIgniter

*Pagination* atau penghalaman merupakan salah satu fitur wajib yang harus dimiliki sebuah aplikasi *web*. Dengan adanya *pagination*, aplikasi *web* Anda tidak perlu menampilkan seluruh data dari tabel yang diambil datanya. Cukup menampilkan beberapa data kemudian jika *user* ingin melihat lagi data selanjutnya tinggal memilih halaman yang dihasilkan oleh aplikasi *web* Anda.

Untuk itu kita akan mengenal *pagination* sederhana di CodeIgniter. Dengan menggunakan data dari tabel **user**, kita akan mencoba menampilkan sebagian data dari tabel tersebut dan akan mengganti data yang diambil lagi ketika mengklik halaman baru. Untuk itu kita buat terlebih dahulu *controller* yang bernama **paginationssample** kemudian taruh di **application -> controllers**.

```
<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Paginationsample
extends CI_Controller {
    public function    construct()
    {
        parent::    construct();
        $this->load->helper('url');
        $this->load->library('input');
        $this->load->model('daftaragenda/agenda_model');
    }

    public function index($offset=0)
    {
        // tentukan jumlah data per halaman
        $perpage = 3;

        // load library pagination
        $this->load->library('pagination');

        // konfigurasi tampilan paging
        $config = array(
            'base_url' => site_url('paginationssample/index'),
            'total_rows' => count($this->agenda_model-
                >select_all()->result()), 'per_page' =>
                $perpage,
        );

        // inisialisasi pagination dan config
        $this->pagination->initialize($config);
        $limit['perpage'] = $perpage;
        $limit['offset'] = $offset;

        $data['daftar_agenda'] = $this->agenda_model->select_all_paging($limit)-
            >result();
        $this->load->view('paginationssample/daftar_agenda_paging', $data);
    }
}
/* End of file welcome.php */
/* Location: ./application/controllers/welcome.php */
```

Pada kode diatas, kita membutuhkan *helper url*, *library input*, dan *model agenda\_model*. Pertama, kita menentukan jumlah *item* yang ingin ditampilkan setiap halaman. Kemudian kita gunakan *library pagination*. Kemudian kita konfigurasi *pagination* yang akan kita hasilkan. Kita memilih *function* apa yang akan menggunakan *pagination*, mencatat total data yang akan ditampilkan, dan menentukan banyak *item* yang akan ditampilkan. Konfigurasi kemudian digunakan oleh *library pagination*. Dan kita melewati batas awal (*offset*) dan banyaknya *item* yang akan ditampilkan untuk *pagination* ke *function select\_all\_paging()* di *model agenda\_model*.

Dengan masih menggunakan *model agenda\_model* tambahkan *function select\_all\_paging()* di *model* tersebut. Berikut adalah *source code* dari *function select\_all\_paging()*:

```
<?php
class Agenda_model extends CI_Model {
    .....
    .....
    .....
    // function yang digunakan oleh
    paginationsample function
    select_all_paging($limit=array()){
        $this->db->select('*');
        $this->db->from('agenda');
        $this->db->order_by('date_modified', 'desc');

        if ($limit != NULL)
            $this->db->limit($limit['perpage'], $limit['offset']);

        return $this->db->get();
    }
}
```

Pada *function select\_all\_paging()* diatas, data yang diambil dibatas sesuai *offset* dan banyaknya *item* yang ingin ditampilkan di *function index* di *controller paginationsample*. Sekarang kita akan melihat *source code* dari *view* untuk menampilkan *pagination* yang bernama **daftar\_agenda\_pagination**.

```
<!DOCTYPE html>
<html>
<head>
<title>Daftar Hadir Praktikum</title>
</head>
<body>
<h2>Daftar Agenda</h2>
<a href="<?php echo site_url('daftarhadir/tambah_agenda');?>">Tambah Agenda</a>
<br />
<br />

<br />
<?php echo $this->pagination->create_links(); ?>
<br />
<br />

<?php foreach ($daftar_agenda as $agenda) {?>
<fieldset>
```

```

<h3><?php echo $agenda->nama;?></h3>
<a href="<?php echo site_url('daftarhadir/edit_agenda/' . $agenda-
>id_agenda);?>">Edit</a> |
<a href="<?php echo site_url('daftarhadir/delete_agenda/' . $agenda-
>id_agenda);?>">Delete</a>
<br />
<p>
<?php echo $agenda->keterangan;?>
</p>
</fieldset>
<br />
<?php } ?>

<br />
<?php echo $this->pagination->create_links(); ?>
<br />
<br />
</body>
</html>

```

Sebelumnya buat terlebih dahulu *folder* dengan nama **paginationsample** di **application -> views**. Kemudian simpan *file* diatas dengan nama **daftar\_agenda\_pagination.php** di dalam *folder* **paginationsample**. Kemudian akses *controller* **paginationsample** dengan URL berikut:

<http://localhost/pelatihanbasdat/index.php/paginationsample>

Kemudian Anda akan melihat tampilan seperti berikut:



## 10.4 Memahami Form Handling di CodeIgniter

### 10.4.1 Membuat Form dengan Helper Form

Di CodeIgniter terdapat sebuah fitur yang digunakan untuk menangani data yang dikirimkan oleh *user* melalui *form*. Anda cukup menentukan *rule* untuk proses validasi yang akhirnya dapat mengurangi tenaga untuk membuat logika validasi sendiri. Jika validasi gagal, CodeIgniter akan menampilkan *error* dari validasi tersebut ke halaman *form* dimana Anda mengirimkan data Anda.

Bahkan Anda dapat menggunakan fitur untuk menghasilkan *field* yang lebih cepat dengan menggunakan *library form* di CodeIgniter. Jika biasanya Anda mengetikkan sintaks HTML secara penuh untuk mendefinisikan sebuah *text field* maka dengan menggunakan *function form\_input()* *helper form*. Anda dapat meringkas proses penulisan *field* di *form*.

Kita akan mencoba bagaimana membangun *form* dengan *helper form* dan mencoba memberikan validasi pada *form* tersebut dengan *library form\_validation*. Berikut adalah *controller* yang akan

```
<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Formhandling extends
CI_Controller {
    public function    construct()
    {
        parent::    construct();
        $this->load->helper('url');
        $this->load->helper('form');
        $this->load->library('input');
        $this->load->library('form_validation');
    }

    public function index()
    {
        $this->load->view('formhandling/form_register_user');
    }

    . . . . .
    . . . . .
    . . . . .
}
/* End of file formhandling.php */
/* Location: ./application/controllers/formhandling.php */
```

digunakan untuk eksperimen kita kali ini:

Sebelumnya simpan *source code* diatas dengan nama **formhandling.php** kemudian simpan di *folder application* → **controllers**. Kemudian mari kita coba untuk membuat *view* yang *field* di *form*-nya dibuat menggunakan *helper form*. Berikut adalah *source code* yang akan ditulis:

```

<html>
<head>
  <title>Form Handling</title>
</head>
<body>
  <h1>Ayo Gabung Bersama Kami</h1>
  <fieldset>

    <?php echo form_open('formhandling/proses_register_user');?>
    <?php echo form_error('nama', '<div
    style="color:red">', '</div>');?> Nama : <?php echo
    form_input('nama');?> <br /><br/>
    <?php echo form_error('samaran', '<div
    style="color:red">', '</div>');?> Samaran : <?php
    echo form_input('samaran');?> <br /><br/>
    <?php echo form_error('email', '<div
    style="color:red">', '</div>');?> Email : <?php
    echo form_input('email');?> <br /><br/>
    <?php echo form_error('password', '<div
    style="color:red">', '</div>');?> Password : <?php
    echo form_password('password');?> <br /><br/>
    <?php echo form_error('ulangpassword', '<div
    style="color:red">', '</div>');?> Ulang Password : <?php
    echo form_password('ulangpassword');?> <br /><br/>
    <?php echo form_error('umur', '<div
    style="color:red">', '</div>');?> Umur : <?php
    echo form_input('umur');?> <br /><br/>
    Twitter : <?php echo form_input('twitter');?>
    <br /><br/> website : <?php echo
    form_input('website');?> <br /><br/>
    <?php echo form_submit('daftar', 'Daftarkan Saya !');?>
    <?php echo form_close(); ?>
  </fieldset>
</body>
</html>

```

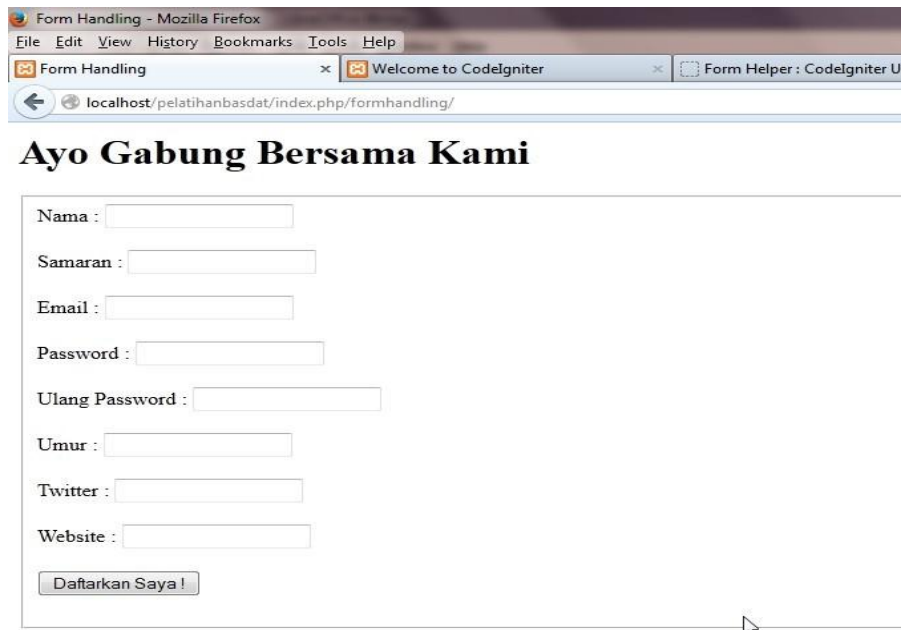
Beri nama *file* dari *source code* diatas dengan nama **form\_register\_user.php**. Sebelumnya buat terlebih dahulu *folder* yang bernama **formhandling** di *folder application* → **views**. Dan simpanlah *source code* diatas di *folder formhandling*. Beberapa *function* dari *helper form* yang harus diingat antara lain:

- **form\_open()**, *function* yang digunakan untuk membuka *form* dan menentukan tipe *form*.
- **form\_error()**, *function* yang digunakan untuk memperlihatkan *error* di *form* ketika proses validasi
- **form\_input()**, *function* yang digunakan untuk menyingkat penulisan *input* tipe *text* di *form*
- **form\_password()**, *function* yang digunakan untuk menyingkat penulisan *input* tipe *password* di *form*
- **form\_submit()**, *function* yang digunakan untuk menyingkat penulisan tombol *submit* di *form*
- **form\_close()**, *function* yang digunakan untuk menutup *form*

Dengan menggunakan *helper form* tersebut, setidaknya kita dapat menghemat waktu untuk membuat *form* di *aplikasi web* yang akan kita bangun. Mari kita coba lihat *form pendaftaran* diatas lewat URL berikut:

<http://localhost/pelatihanbasdat/index.php/formhandling/>

Dan cobalah lihat dengan mode *view source code* pada *browser* yang Anda gunakan. Sintaks untuk membuat *form* sudah ditangani oleh CodeIgniter. Berikut adalah *screenshot* dari *form* yang diciptakan oleh CodeIgniter:



The screenshot shows a web browser window with the title 'Form Handling - Mozilla Firefox'. The address bar displays 'localhost/pelatihanbasdat/index.php/formhandling/'. The page content features a heading 'Ayo Gabung Bersama Kami' and a registration form. The form contains the following fields: 'Nama :', 'Samaran :', 'Email :', 'Password :', 'Ulang Password :', 'Umur :', 'Twitter :', and 'Website :'. Each field is followed by a text input box. Below these fields is a button labeled 'Daftarkan Saya!'. The browser's tab bar shows three tabs: 'Form Handling', 'Welcome to CodeIgniter', and 'Form Helper : CodeIgniter Us'.

### 3.2. Memberikan Proses Validasi dengan Library Form Validations

*Form* diatas belum memiliki penanganan ketika tombol *submit* ditekan. Untuk itu kita akan membuat *function* baru yang memanfaatkan *library form\_validation* untuk menangani proses validasi dari *form* yang telah kita buat. Berikut adalah *source code* dari *function*

```
<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Formhandling extends
CI_Controller {

. . . . .

public function proses_register_user(){
    $this->form_validation->set_rules('nama', 'Nama', 'required');
    $this->form_validation->set_rules('samaran', 'Samaran', 'required');
    $this->form_validation->set_rules('password', 'Password', 'required');
    $this->form_validation->set_rules('ulangpassword', 'Ulang Password',
    'required');
    $this->form_validation->set_rules('email', 'Email',
    'required|valid_email');
    $this->form_validation->set_rules('umur', 'Umur', 'integer');

    if ($this->form_validation->run() == FALSE)
    {
        $this->load->view('formhandling/form_register_user');
    }
    else
    {
        $this->load->view('formhandling/success_register_user');
    }
}

}
/* End of file formhandling.php */
/* Location: ./application/controllers/formhandling.php */
```

**proses\_register\_user()** di *controller form\_handling* yang akan kita gunakan untuk memvalidasi *form* yang telah dibuat:

Bisa Anda lihat pada *source code* diatas bahwa untuk menentukan aturan validasi cukup dengan menggunakan **form\_validation->set\_rules()** dengan parameter pertama adalah nama dari field yang akan diberikan validasi, parameter kedua adalah nama dari field yang akan ditampilkan dalam peringatan error, dan parameter ketiga adalah jenis validasi yang akan digunakan. Berikut adalah *source code* untuk melihat halaman sukses setelah validasi:

```
<html>
<head>
  <title>Form Handling</title>
</head>
<body>
  <h3>Anda sudah bergabung bersama kami :D</h3>

  <p><?php echo anchor('formhandling', 'Coba Lagi !'); ?></p>
</body>
</html>
```

Simpan *source code* diatas dengan nama **success\_register\_user.php** dan simpan di *folder formhandling* di **application → views**. Untuk menjalankan validasi dari *form* tersebut, digunakanlah **form\_validation->run()** untuk menguji validasi. Jika sukses diarahkan ke halaman sukses. Jika gagal maka akan dikembalikan ke halaman *form* dengan memberikan informasi dari *form* yang *error*. Mari kita coba *form* tersebut dengan kasus tanpa diis

Form Handling - Mozilla Firefox  
File Edit View History Bookmarks Tools Help  
Form Handling x Welcome to CodeIgniter  
localhost/pelatihanbasdat/index.php/formhandling/proses\_register\_user

## Ayo Gabung Bersama Kami

The Nama field is required.  
Nama :

The Samaran field is required.  
Samaran :

The Email field is required.  
Email :

The Password field is required.  
Password :

The Ulang Password field is required.  
Ulang Password :

Umur :

Twitter :

Website :

Daftarkan Saya !

Form Handling - Mozilla Firefox  
File Edit View History Bookmarks Tools Help  
Form Handling x Welcome to CodeIgniter  
localhost/pelatihanbasdat/index.php/formhandling/proses\_register\_user

## Ayo Gabung Bersama Kami

Nama :

Samaran :

The Email field must contain a valid email address.  
Email :

Password :

Ulang Password :

Umur :

Twitter :

Website :

Daftarkan Saya !



Form Handling - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Form Handling x Welcome to CodeIgniter

localhost/pelatihanbasdat/index.php/formhandling/proses\_register\_user

## Ayo Gabung Bersama Kami

Nama :

Samaran :

The Email field must contain a valid email address.

Email :

Password :

Ulang Password :

Umur :

Twitter :

Website :

*Gambar 7.5 Form Ketika Diisi Dengan Data yang Sesuai*

Form Handling - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Form Handling x Welcome to CodeIgniter

localhost/pelatihanbasdat/index.php/formhandling/proses\_register\_user

**Anda sudah bergabung bersama kami :D**

[Coba Lagi !](#)

## 4. Membuat Fitur Upload di CodeIgniter

### 4.1. Membuat Form Upload File

Fitur *upload* diperlukan untuk mencatat informasi dari *user* yang menyertakan *file*. *File* tersebut bisa berupa gambar, suara, video, program, dokumen, *slide*, atau *file* lainnya. Untuk itu jenis *form* yang dibutuhkan pun berbeda. Sekarang kita akan membuat *controller* yang menampilkan *form upload* dahulu. Berikut adalah *source code* nya:

```
<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class uploadsample extends
CI_Controller {
    public function    construct()
    {
        parent::    construct();
        $this->load->helper('url');
        $this->load->helper('form');
        $this->load->library('input');
    }

    public function index()
    {
        $this->load->view('uploadsample/form_upload', array('error'=>''));
    }

    .....
}
/* End of file uploadsample.php */
/* Location: ./application/controllers/uploadsample.php */
```

Simpanlah *source code* diatas dengan nama **uploadsample.php** kemudian simpan di **application -> controllers**. Seperti yang tertulis di *source code*, *controller* ini memerlukan dua buah *helper* yaitu **url** dan **form** serta membutuhkan *library* **input**. Disana terdapat sebuah *function* **index()** yang akan menampilkan *form upload* yang akan kita tulis.

*Untuk form upload sendiri yang akan digunakan dalam percobaan ini adalah sebagai berikut:*

```
<html>
<head>
  <title>Demo Upload File</title>
</head>
<body>

<?php echo $error;?>

<?php echo
form_open_multipart('uploadsample/proses_upload');
?> Judul: <br />
```

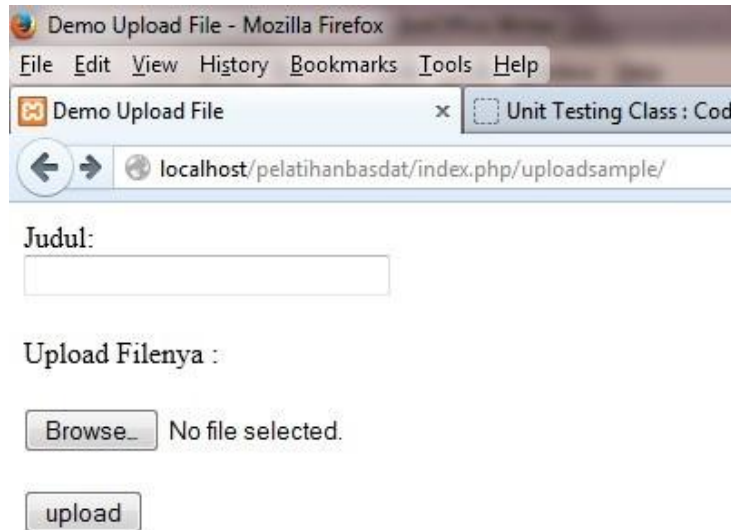
```
<input type="text" name="judul" size =
"30" /> <br/><br/> Upload Filenya : <br
/><br />
<input type="file" name="userfile" size="20" />
<br /><br />
<input type="submit" value="upload" />
</form>

</body>
</html>
```

Ingat untuk membuat sebuah *form* yang menyertakan *uploading* digunakanlah **form\_open\_multipart()** agar *form* tersebut dapat meng-*upload file* yang kita sertakan di *form*. Sebelum disimpan, buatlah terlebih dahulu *folder* yang bernama **uploadsample** di **application->views**. Kemudian simpan *source code* diatas dengan nama **form\_upload.php** dan simpan di *folder* **uploadsample**.

*Untuk melihat wujud dari view diatas akseslah lewat URL berikut ini:*

<http://localhost/pelatihanbasdat/index.php/uploadsample/>



## 4.2. Memproses File yang Telah Di-upload

Sekarang kita akan mencoba menulis untuk membuat kode yang memproses *uploading file* dari *form* yang telah kita buat sebelumnya. Sebelumnya buat terlebih dahulu *folder* yang akan menyimpan *file* hasil *upload* dengan nama **upload** dan simpan di *root* projek tepatnya di *folder pelatihanbasdat*. Kemudian buatlah *function* yang bernama **proses\_upload()** di *controller uploadsample*. Berikut adalah *source code* yang harus ditambahkan di *controller uploadsample*:

```

<?php if ( ! defined('BASEPATH')) exit('No direct
script access allowed'); class Uploadsample extends
CI_Controller {

. . . . .

. . . . .

        public function
        proses_upload(){
        $judul = $this->input->post('judul');
        $config['upload_path'] = './upload/';
        $config['allowed_types'] = 'gif|jpg|png';
        $config['max_size'] = '100';
        $config['max_width'] = '1024';
        $config['max_height'] = '768';

        $this->load-

        >library('upload',

        $config); if (!$this-

        >upload->do_upload()){
            $error = array('error'=>$this->upload->display_errors());
            $this->load->view('uploadsample/form_upload', $error);
        }
        else {
            $upload_data = $this->upload->data();
            $upload_data['judul'] = $judul;
            $data = array('upload_data' => $upload_data);
            $this->load->view('uploadsample/view_upload_success', $data);
        }
    }
}

```

Pada *source code* diatas, Anda dapat melihat beberapa *config* dasar yang dibutuhkan untuk proses *upload file*. Di dalam *config* tersebut terdapat:

- **upload\_path** untuk menentukan letak *folder* yang akan menyimpan *file* yang di-*upload*
- **allowed\_types** untuk menentukan jenis *file* apa saja yang boleh di-*upload*
- **max\_size** untuk menentukan ukuran maksimal *file* yang boleh di-*upload*
- **max\_width** untuk menentukan lebar maksimal *file* yang boleh di-*upload*
- **max\_height** untuk menentukan tinggi maksimal *file* yang boleh di-*upload*

Setelah itu, *config* yang telah ditentukan akan dilewatkan kedalam proses pemanggilan *library upload* sebelum melakukan proses *upload*. Untuk memulai proses *upload* dilakukan dengan memanggil perintah **upload->do\_upload()**. Jika berhasil maka Anda dapat memanggil **upload->data()** untuk mendapatkan data hasil *upload* yang mungkin dapat Anda simpan di *database*. Sedangkan jika gagal, maka proses *upload* akan memberitahu kesalahan apa yang terjadi selama proses *upload* dan memperlihatkan *error* tersebut di *form upload*.

Sebelum mencoba form upload tersebut mari kita buat terlebih dahulu halaman sukses yang diperlukan untuk menampilkan data – data dari file hasil upload:

```

<html>
<head>
<title>Demo Upload File</title>
</head>
<body>
<h3>Proses Upload Berhasil !!!</h3>

<ul>
<?php foreach ($upload_data as $item => $value):?>
<li><?php echo $item;?>: <?php echo $value;?></li>
<?php endforeach; ?>
</ul>

<p><?php echo anchor('uploadsample', 'Coba lagi !'); ?></p>
</body>
</html>

```

Simpan *source code* diatas dengan nama **view\_upload\_success.php** dan simpan di *folder* **uploadsample**.

