MODUL IX-FRAMEWORK CI (BAGIAN 1)

9.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami cara kerja sebuah framework.
2. Memahami cara instalasi framework CodeIgniter.
3. Mampu membuat aplikasi CRUD menggunakan framework CodeIgniter.

9.2 Pengantar Framework CodeIgniter

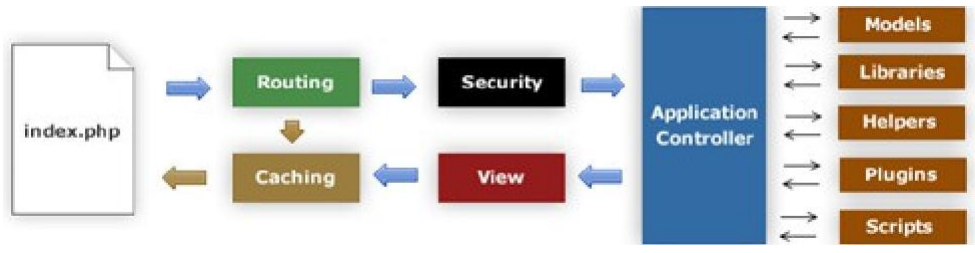
CodeIgniter adalah sebuah web framework yang dikembangkan oleh Rick Ellis dari Ellis Lab. CodeIgniter dirancang untuk menjadi sebuah web framework yang ringan dan mudah untuk digunakan. Bahkan pengakuan dari Rasmus Lerdorf, pencipta bahasa pemrograman PHP, mengatakan bahwa CodeIgniter merupakan web framework mudah dan handal.

Sebelum mencoba CodeIgniter, perlu diketahui istilah *web framework* itu sendiri. Menurut Microsoft Computer Dictionary, *web* adalah sekumpulan dokumen yang saling terhubung dalam sistem *hypertext* yang penggunanya akan menjelajahi *web* melalui halaman beranda. Sedangkan *framework* adalah desain struktur dasar yang dapatd digundakan kembali *(reusebale)* yang terdiri dari *abstract class* dan *concrete class* di pemrograman berorientasi objek.

Menurut dokumentasi CodeIgniter, CodeIgniter merupakan *toolkit* bagi orang yang ingin membangun aplikasi *web* menggunakan PHP. Tujuannya adalah membuat pengembangan proyek menjadi lebidh cepat dibandingkan dengan menulis kode dari awal (*scratch*). CodeIgniter menyediakan kumpulan *library* untuk tugas – tugas yang sering dilakukan (*commonly needed task*) dan sangat mudah undtuk mengakses *library* yang tersedia di CodeIgniter. Dengan menggunakan CodeIgniter, kita cukup fokus pada pengembangan projek dan meminimalisir jumlah kode yang akan ditulis.

9.3 Cara Kerja CodeIgniter

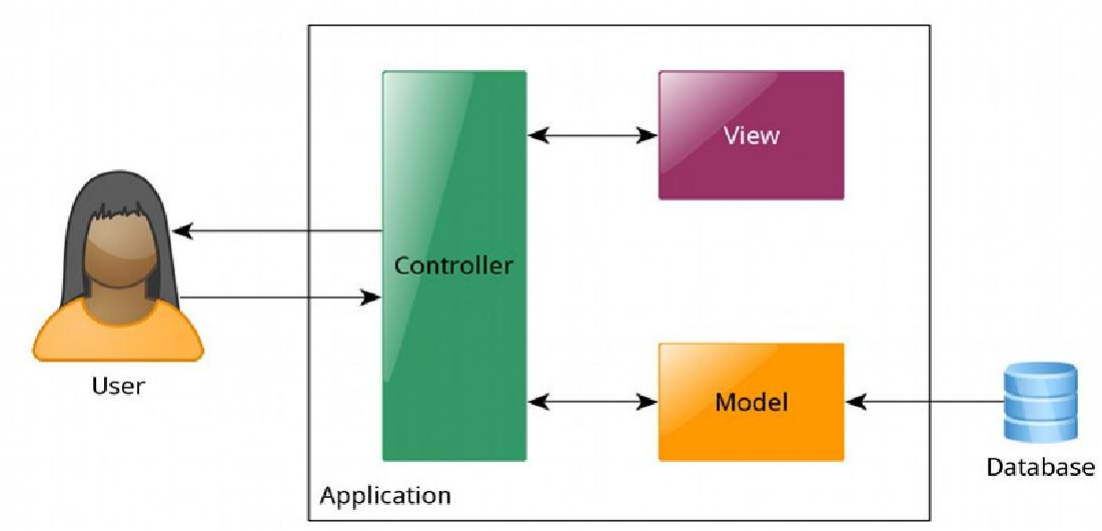
Untuk melengkapi pemahaman mengenai CodeIgniter, berikut terdapat sebuah diagram yang menjelaskan bagaimana CodeIgniter bekerja:



Berikut adalah penjelasan cara kerja Code Igniter:

1. index.php bertindak sebagai controller terdepan, dan menginisialisasi resource yang diperlukan untuk menjalankan Code Igniter
2. Router memeriksa HTTP request untuk menentukan apa yang harus dikerjakan
3. *Jika cache file ada, maka akan ditampilkan langsung, dengan melewati eksekusi normal sistem*
4. Sebelum memuat controller, HTTP request akan memeriksa apa yang disubmit user dan memfilternya untuk keamanan
5. Controller memuat model, core libraries, plugin, helper, dan resource lainnya untuk memproses permintaan tertentu.
6. View ditampilkan di browser sesuai proses yang dikerjakan controller.

9.4 Cara Kerja MVC



CodeIgniter menggunakan pendekatan *Model-View-Controller*, yang bertujuan untuk memisahkan logika dan presentasi. Konsep ini mempunyai keunggulan dimana desainer dapat bekerja pada *template file*, sehingga redundansi kode presentasi dapat diperkecil. Berikut adalah konsep *Model-View-Controller* yang diterapkan di CodeIgniter:

1. *Model* menggambarkan struktur data. Biasanya kelas model akan berisi fungsi yang digunakan untuk mengambil, menambah, dan memperbaharui informasi yang ada di database.
2. *View* adalah informasi yang diperlihatkan kepada user. View adalah halaman web yang terdiri dari HTML, CSS dan Javascript, tapi pada Code Igniter, view dapat juga sebagai potongan halaman seperti header atau footer. Bahkan dapat juga halaman RSS atau tipe halaman lainnya.
3. *Controller* adalah perantara Model, View, dan resource lainya yang dibutuhkan untuk menangani HTTP request dan menghasilkan halaman web.

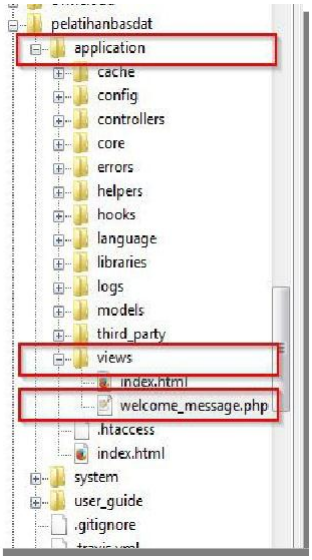
9.5 Instalasi CodeIgniter

Tentunya untuk membuat aplikasi web *dengan CodeIgniter, kita harus mengenal cara instalasinya terlebih dahulu. Berikut adalah cara untuk instalasi CodeIgniter di Windows 7:*

1. unduh CodeIgniter dari *link* berikut : https://codeigniter.com/download
2. kemudian ekstrak bundelan **CodeIgniter_2.1.4** di tempat unduhan Anda
3. copy *hasil ekstraksi ke* folder *C:/xampp/htdocs atau direktori xampp di mesin Anda*
4. ubah namanya menjadi **pelatihanbasdat**
5. *akses direktori tersebut lewat* browser *dengan URL : http://localhost/pelatihanbasdat*

Di CodeIgniter terdapat hirarki yang dikepalai oleh tiga *folder* utama, yaitu : **application, system,** dan **user_guide**.

9.6 Mengubah Tampilan Awal CodeIgniter



Sebagai percobaan kita akan mencoba mengubah halaman awal CodeIgniter, untuk melihat bagaimana CodeIgniter bekerja. Siapkanlah *Text Editor* favorit Anda yang akan digunakan untuk menyunting beberapa *source code* yang terdapat di CodeIgniter.

Sebelum menyunting *view* **welcome_message.php**. Kita harus mengubah **$config['base_url']** yang berada di **application -> config -> config.php** dari http://www.example.com menjadi http://localhost/pelatihanbasdat . Hal tersebut dilakukan agar nama domain yang digunakan dapat digunakan di seluruh bagian kode program yang ada di CodeIgniter.

Berikut adalah langkah – langkah untuk mengubah tampilan halaman awal CodeIgniter. Buka *Text Editor* yang sering Anda gunakan. Cari *file* **welcome_message.php** yang terdapat di direktori **pelatihanbasdat->application->views.** Kemudian buka *file* tersebut menggunakan *Text Editor*, Kemudian Anda cari bagian *source code* seperti pada *listing berikut:*

```
.....................................................................


<div id="container">
	<h1>Welcome to CodeIgniter!</h1>

	<div id="body">
		<p>The page you are looking at is being generated dynamically by
		CodeIgniter.</p>

		<p>If you would like to edit this page you'll find it located at:</p>
		<code>application/views/welcome_message.php</code>

		<p>The corresponding controller for this page is found at:</p>
		<code>application/controllers/welcome.php</code>

		<p>If you are exploring CodeIgniter for the very first time, you should
start by reading the <a href="user_guide/">User Guide</a>.</p>
	</div>

	<p class="footer">Page rendered in <strong>{elapsed_time}</strong>
	seconds</p>
</div>


.....................................................................
```

9.7 Membuat Controller Baru

Apabila tadi kita hanya menyunting *controller* **welcome** yang sudah tersedia di CodeIgniter, sekarang kita akan mencoba untuk membuat *controller* baru yang bernama **calculator**. *Controller* tersebut digunakan untuk menampilkan *form* kalkulator dan menampilkan hasil perhitungan yang ditentukan oleh *user*.

Pertama, buatlah *controller* baru di *folder* **application** -> **controllers** dengan nama **calculator**. Di dalam *controller* ini terdapat tiga *function* yang akan ditulis, pertama adalah *constructor* yang berfungsi untuk memuat *helper* dan *library*, kedua adalah *function* **index** yang digunakan untuk menampilkan *form* kalkulator, dan yang ketiga adalah *function* untuk memproses perhitungan yang diinginkan oleh *user*.

Sebelum menyelesaikannya, mari kita salin *function* yang pertama dan kedua. Berikut adalah kode sumber *function* **index** dan *constructor*:

```
<?php if ( ! defined('BASEPATH')) exit('No direct

script access allowed'); class Calculator extends

CI_Controller {

  public function    construct()
  {
    parent::   construct();
    $this->load->helper('url');
    $this->load->library('input');
  }

  public function index()
  {
    $this->load->view('form_hitung');
  }

. . . . . .
. . . . . .
. . . . . .

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

Kemudian setelah kita menyalin *source code* diatas, sekarang kita buat *view* **form_hitung** di **application – views** yang akan ditampilkan ketika *controller* **calculator** diakses oleh user. Berikut adalah *source code* **form_hitung** yang akan digunakan oleh *controller* **calculator**:

```
<html>
  <head>
    <title>Kalkulator CI</title>

  </head>
  <body>
    <h1>Kalkulator CI</h1>
    <form action="<?php echo site_url('calculator/hasil_hitung');?>" method="POST">
      <input type="text" name="angka1"/> <br /><br/>
      <input type="text" name="angka2"/> <br /><br/>
      <select name="pilih-hitung">
        <option value="+">+</option>
        <option value="-">-</option>
        <option value="*">*</option>
        <option value="/">/</option>
      </select><br/><br/>
      <input type="submit" value="Hitung" />
    </form>
  </body>
</html>
```

View diatas menerima dua masukan yaitu **angka1** dan **angka2**. Kemudian *user* akan memilih operasi yang disediakan oleh **calculator**. Kemudian masukan tersebut dikirim ke *function* **hasil_hitung** di *controller* **calculator.** Untuk melihat *form* diatas akses *view* tersebut melalui URL berikut :

```
http://pelatihanbasdat/index.php/calculator
```

Setelah Anda mengakses URL diatas, tampilan pada gambar 3.1 disamping akan tampak pada *browser* Anda.

Dengan demikian tinggal menulis *function* **hasil_hitung()**. Pada *function* tersebut akan terjadi proses penangkapan nilai – nilai dari *form*, penentuan proses perhitungan sesuai yang ditentukan oleh *user*, membungkus hasil perhitungan, kemudian menampilkannya di *view* **hasil_hitung**.

Sebelum menulis *source code view* **hasil_hitung**, mari kita tulis dulu *source code* dari *function* **hasil_hitung()**. *Source code* tersebut adalah sebagai berikut:

```
<?php if ( ! defined('BASEPATH')) exit('No direct

script access allowed'); class Calculator extends

CI_Controller {

. . . . . .
. . . . . .
. . . . . .

     public function
     hasil_hitung(){
    // mengecek masukan dari form
    $angka1 = $this->input->post('angka1');
    $angka2 = $this->input->post('angka2');
    $pilih_hitung = $this->input->post('pilih-hitung');

    $hasil_hitung = 0;
    // mengecek proses perhitungan
    yang diminta if ($pilih_hitung
    == "+"){
 $hasil_hitung = $angka1 + $angka2;
    }
    else if ($pilih_hitung == "-"){
 $hasil_hitung = $angka1 - $angka2;
    }
    else if ($pilih_hitung == "*"){
 $hasil_hitung = $angka1 * $angka2;
    }
    else if ($pilih_hitung == "/"){
 $hasil_hitung = $angka1 / $angka2;
    }

    // membungkus semua data perhitungan untuk ditampilkan di view
    $data['angka1'] = $angka1;
    $data['angka2'] = $angka2;
    $data['pilih_hitung'] = $pilih_hitung;
    $data['hasil_hitung'] = $hasil_hitung;

    // menampilkan hasil
    $this->load->view('hasil_hitung', $data);

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

Sedangkan untuk *source code view* **hasil_hitung** adalah sebagai berikut:

```
<h1> Hasil perhitungan </h1>

<h3>
  <?php echo $angka1." $pilih_hitung ".$angka2." = ".$hasil_hitung?>
</h3>

<a href="<?php echo site_url('calculator/');?>"><< Kembali menghitung</a>
```

View *diatas menampilkan hasil perhitungan dan menu untuk kembali menghitung. Berikut adalah hasil tangkapan layar dari* view ***hasil_hitung****:*

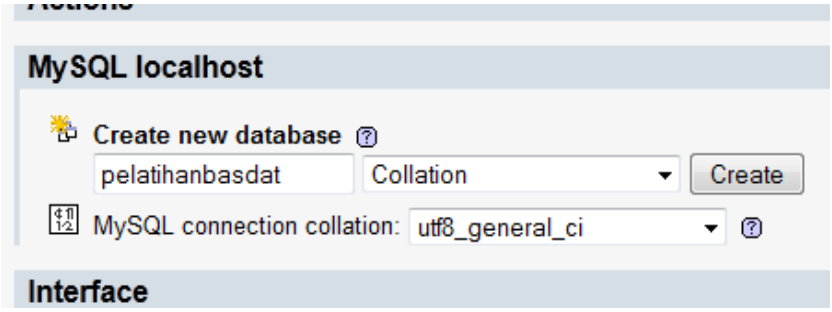
9.8 Mengenal Create Read Update Delete di CodeIgniter

9.8.1 Membuat Database Agenda

Dalam membangun sebuah aplikasi tentu kaidah *create, read, update, delete* (CRUD) merupakan sebuah kewajiban dasar yang tentunya diperlukan juga di aplikasi *web*. Dengan kaidah tersebut *user* dapat memberikan masukan data baru, melihat data, memperbaharui data, atau menghapus salah satu data. Untuk membuat aplikasi tersebut, tentu salah satu hal yang dibutuhkan adalah ketersediaan *storage* untuk menyimpan perubahan data. *Storage* tersebut dapat berupa *file* atau *database* yang siap menampung perubahan data yang dipengaruhi oleh *user*.

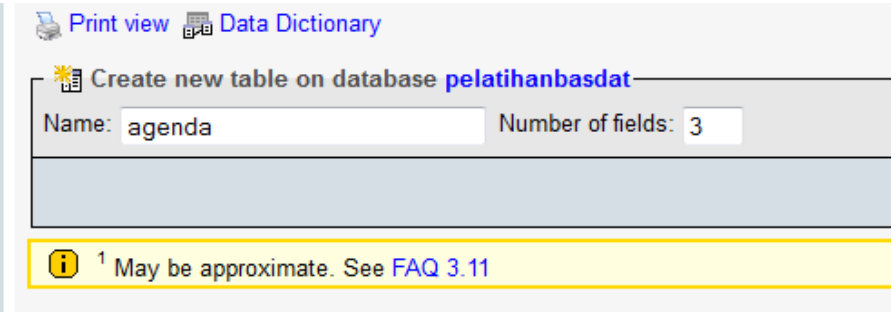
Dalam praktikum kali ini dibutuhkan MySQL sebagai sarana penyimpanan data yang akan digunakan oleh *user*. Kita akan membuat sebuah aplikasi yang mencatat agenda. Kita dapat mengisikan agenda baru, memperbaharui agenda yang sudah ada, melihat daftar agenda yang pernah dicatat, kemudian menghapus agenda yang sudah terlewat atau yang sudah tidak diperlukan.

Bagaimanapun sebelum membuat tabel, tentu harus membuat *database* terlebih dahulu. Kita namai *database* yang akan digunakan adalah **pelatihanbasdat**. Di halaman muka PHPMyAdmin ketikkan nama tersebut dibawah *field* **Create new database**, kemudian tekan tombol **Create** untuk membuat database yang diinginkan.

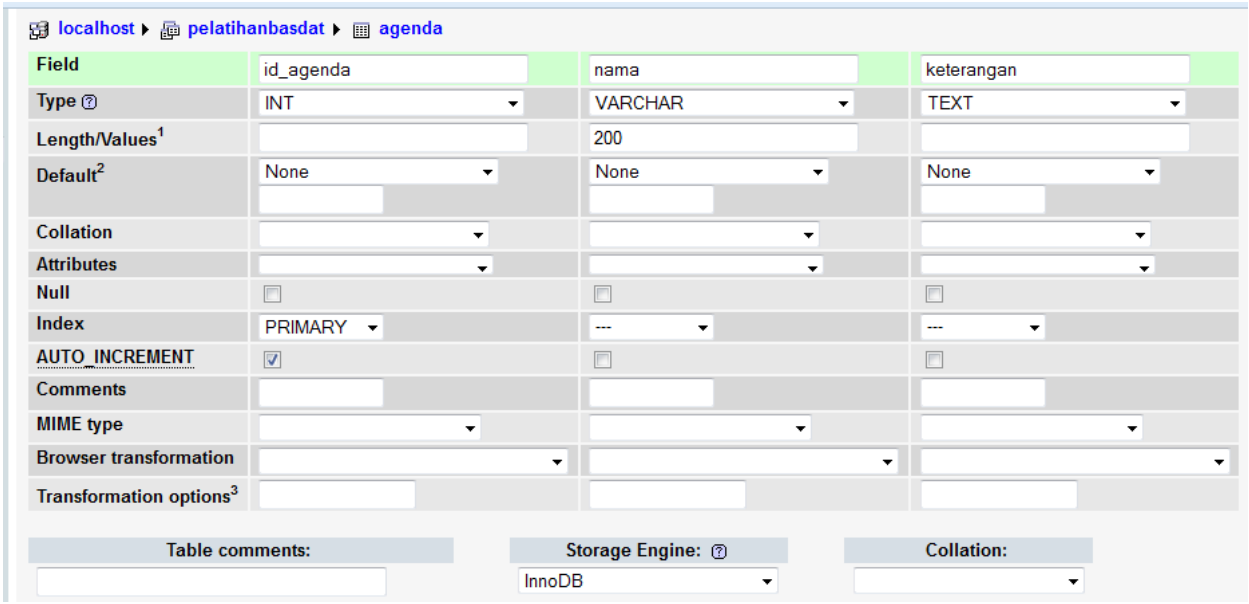


Gambar 4.1 Membuat Database

Saat ini kita hanya selesai membuat *database* saja, tabel yang akan digunakan oleh aplikasi *web* yang akan kita gunakan adalah tabel **agenda**. Untuk membuatnya klik terlebih dahulu *database* yang telah dibuat di panel sebelah kiri halaman muka jika baru masuk ke PHPMyAdmin, atau dapat meneruskan proses sehabis pembuatan *database* jika belum keluar dari halaman PHPMyAdmin. Untuk itu ketikkan nama tabel yaitu, **agenda**, dan jumlah kolom atau *field* yang dibutuhkan. Untuk kasus ini diperlukan tiga buah kolom pada tabel **agenda**.

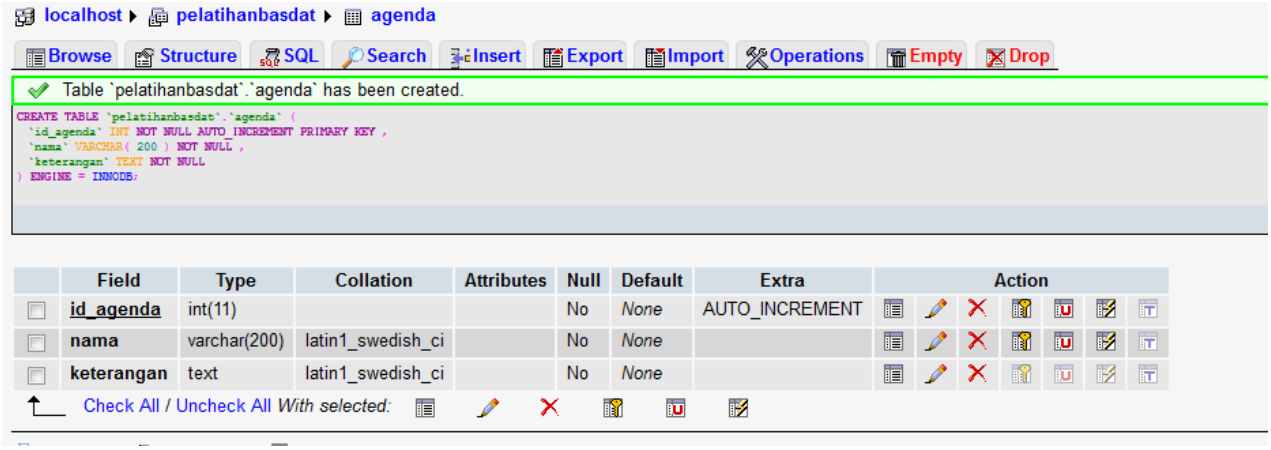


Kemudian Anda akan dibawa ke halaman untuk mengisi kolom sesuai kolom yang dibutuhkan. Pada fase ini kita akan membuat tiga buah *field* yaitu **id_agenda**, **nama**, **keterangan**. Ketiga *field* tersebut mempunyai spesifikasi tersendiri. *Field* **id_agenda** memiliki *type* **int,** dianggap sebagai *primary key*, dan bersifat *auto increment*. *Field* **nama** memiliki *type* **varchar** dan memiliki *length* 200. *Field* **keterangan** memiliki *type* **varchar** dan memiliki *length 200*. Perhatikan gambar dibawah ini:



Setelah selesai mengisikan *field* yang dibutuhkan, tekan tombol **Save** yang berada di bawah *form* pengisian *field – field* tabel:

Jika berhasil, PHPMyAdmin akan menampilkan hasil tabel yang telah berhasil dibuat. Setelah tabel selesai dibuat, Anda dapat mengisikan data – data yang dibutuhkan sebagai data awal, atau membangun aplikasi terlebih dahulu, kemudian mengisikan data lewat aplikasi *web* yang Anda bangun. Berikut adalah tangkapan layar dari tabel yang berhasil dibuat:



Jangan lupa untuk menambahkan data *dummy* pada tabel tersebut karena kita akan mencoba untuk menampilkan data yang ada di dalam tabel **agenda**. Isilah kira – kira 3 ~ 5 data pada tabel tersebut.

9.8.2 Konfigurasi Database di CodeIgniter

Akhirnya *database* yang kita bangun untuk pelatihan ini berhasil dan tabel untuk aplikasi agenda yang akan kita bangun pun sudah berhasil dibuat. Tapi untuk mengaksesnya lewat CodeIgniter, ada beberapa langkah yang harus dilakukan terlebih dahulu.

Kita harus mengkonfigurasi *database* yang akan kita akses lewat CodeIgniter lewat *file* **database.php** yang berada di **application -> config**. Konfigurasi yang akan digunakan dalam pelatihan ini adalah:

```
. . . . . .

|

| The $active_record variables lets you determine whether or not to load

| the active record class

*/

$active_group = 'default';

$active_record = TRUE;

$db['default']['hostname'] = 'localhost';

$db['default']['username'] = 'root';
```

```
$db['default']['cachedir'] = '';

$db['default']['char_set'] = 'utf8';

$db['default']['dbcollat'] = 'utf8_general_ci';

$db['default']['swap_pre'] = '';

$db['default']['autoinit'] = TRUE;

$db['default']['stricton'] = FALSE;
```

Kemudian setelah mengkonfigurasi database, kita harus menambahkan *library* database di **autoload.php** yang berlokasi di **application -> config** . Hal ini dilakukan agar *library* **database** selalu di load dimanapun di *controller* yang ada di CodeIgniter (Tentunya jika semua *controller* memerlukan *database*). Berikut cara menambahkan *library* **database** kedalam *file* **autoload.php**:

```
. . . . . .

/*

| -------------------------------------------------------------------

|  Auto-load Libraries

| -------------------------------------------------------------------

| These are the classes located in the system/libraries folder

| or in your application/libraries folder.

|

| Prototype:
```

9.8.3 Membuat Model di CodeIgniter

Model dipergunakan untk mengakses *database*. Dengan menggunakan *model*, pembuat aplikasi *web* tidak akan mencampur adukkan kode – kode untuk *database* dan logika bisnis yang digunakan dalam aplikasi *web*. Dengan demikian ketika terjadi perubahan kode untuk mengakses *database* perubahan pada logika bisnis dapat dicegah seminimal mungkin.

Sebuah *model* di CodeIgniter, biasanya merepresentasikan satu tabel di database. Dalam kasus ini karena hanya tabel **agenda** yang digunakan berarti *model* yang akan kita buat adalah **agenda_model.php.** Biasanya didalam sebuah *model* terdapat bagian kode untuk mengambil semua data dari tabel, mengambil salah satu baris data, memperbaharui data di tabel, menghapus data di tabel, dan menambahan data baru.

Untuk melihat contoh nyatanya mari kita lihat isi *file* **agenda_model** berikut ini:

```
<?php

class Agenda_model extends

      CI_Model { function

      construct(){
            parent::   construct();

      }

      function insert_agenda($data){
            $this->db->insert('agenda', $data);
      }

      function select_all(){
            $this->db->select('*');
            $this->db->from('agenda');
            $this->db->order_by('date_modified', 'desc');

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

      function select_by_id($id_agenda){
            $this->db->select('*');
            $this->db->from('agenda');
            $this->db->where('id_agenda', $id_agenda);

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

      function update_agenda($id_agenda, $data){
            $this->db->where('id_agenda', $id_agenda);
            $this->db->update('agenda', $data);
      }

      function delete_agenda($id_agenda){
            $this->db->where('id_agenda', $id_agenda);
            $this->db->delete('agenda');
      }
```

```
	// 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();
	}
}

?>
```

Salin *source code* diatas kemudian simpan di **application->models->daftaragenda** dengan nama **agenda_model.php**. Sebelumnya buat terlebih dahulu *folder* **daftaragenda** di dalam *folder* **application -> models.**

Lebih lengkapnya untuk melihat Active Record atau pembahasan tentang *model* di CodeIgniter, dapat Anda lihat di dokumentasi CodeIgniter bagian *Model* di *General Topics* dan bagian *Database Class* di *Driver Reference.*

9.8.4 Membuat Controller untuk Aplikasi Agenda

Akhirnya kita telah menulis *model* **agenda_model** yang akan dipergunakan di *controller* **daftaragenda**. Di dalam *controller* **daftaragenda**, kita akan menampilkan halaman daftar agenda, menampilkan *form* tambah agenda baru, menampilkan *form editing* agenda, proses menghapus agenda, proses penambahan agenda, dan proses pengubahan agenda.

Sekarang kita akan membuat *controller* **daftaragenda** dengan terlebih dahulu menampilkan daftar agenda yang ada di dalam tabel. *Controller* **daftaragenda** ini membutuhkan *helper* **url** dan *library* **input**. Serta membutuhkan *model* **agenda_model** karena akan mengakses tabel **agenda** di *controller* **daftaragenda** ini.

```
<?php if ( ! defined('BASEPATH')) exit('No direct

script access allowed'); class Daftaragenda extends

CI_Controller {

  public function    construct()
  {
    parent::   construct();
    $this->load->helper('url');
    $this->load->library('input');
    $this->load->model('daftaragenda/agenda_model');
  }
```

```
  // bagian
  pengelolaan agenda
  public function
  index()
  {
    $data['daftar_agenda'] = $this->agenda_model->select_all()->result();
    $this->load->view('daftaragenda/daftar_agenda', $data);
  }

. . . . . .
. . . . . .
. . . . . .

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

Seperti yang bisa Anda lihat, untuk menggunakan *model* di *controller* pertama kali Anda harus memuat *model* **agenda_model** terlebih dahulu. Baru setelah itu Anda dapat menggunakan **agenda_model** di *function* manapun di dalam *controller*. Dalam kasus ini **agenda_model** dipanggil di *function* **index**. Kemudian karena kita ingin mengambil semua isi tabel **agenda** maka *function* **select_all()** yang terdapat di **agenda_model** dipanggil. Untuk proses akhirnya jika Anda ingin mendapatkan semua data maka dapat digunakan **result()** , tapi nilai keluarannya akan sebagai *array object*. Sedangkan jika hanya ingin satu baris saja digunakan **row()**, tapi nilai keluarannya akan sebagai *object.*

Kemudian hasil keluaran tersebut disimpan di *array* **$data** dengan naman **daftar_agenda**. Kemudian lewatkan ke *view* **daftar_agenda** untuk menampilkan agenda yang ada di tabel **agenda**. Sebelumnya buat dulu *folder* **daftaragenda** di dalam **application -> views.** Simpan *view* dibawah ini dengan nama **daftar_agenda.php** di *folder* **application->views->daftaragenda**. Berikut adalah *source code* dari *view* **daftar_agenda**:

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

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

  </body>
</html>
```

Di dalam *source code* diatas terdapat *link* untuk menambahkan agenda baru, menampilkan setiap data di dalam tabel melalui *looping*, serta menambatkan *link* **edit** dan **delete** di setiap *item* tabel **agenda**. Untuk melihat bagaimana *function* **index** bekerja, akseslah lewat URL berikut:

```
http://localhost/pelatihanbasdat/index.php/daftaragenda
```

Kemudian tampilannya kira-kira akan seperti berikut ini:

9.8.5 Menambahkan Fitur Tambah Agenda di Controller Agenda

Jika di subbab sebelumnya kita hanya dapat melihat daftar agenda, maka sekarang kita kan mencoba menambah agenda baru pada aplikasi kita ini. Untuk mewujudkannya, kita memerlukan *form* untuk menerima masukan dari *user*, *function* di *controller* **daftaragenda** untuk menampilkan *form* dan memproses *form* tersebut, serta menggunakan *function* yang dapat menambahkan data baru di **agenda_model**. Masih di *file controller* yang sama yaitu **daftaragenda.php**, tambahkan potongan kode berikut:

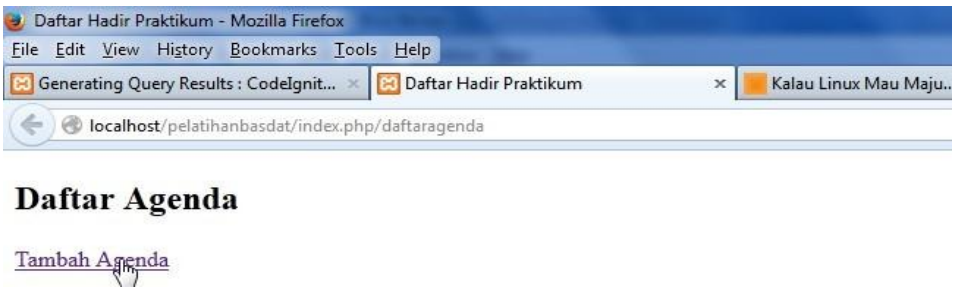
```
<?php if ( ! defined('BASEPATH')) exit('No direct

script access allowed'); class Daftaragenda extends

. . . . . .
. . . . . .
. . . . . .

  public function tambah_agenda(){
    $this->load->view('daftaragenda/form_tambah_agenda');
  }

  public function proses_tambah_agenda(){
    $data['nama'] = $this->input->post('nama');
    $data['keterangan'] = $this->input->post('keterangan');
    $this->agenda_model-
    >insert_agenda($data);
    redirect(site_url('daftaragenda'
    ));
  }

. . . . . .
. . . . . .
. . . . . .

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

Pada kode diatas *function* **tambah_agenda()** digunakan untuk menampilkan *view* **form_tambah_agenda**. Kemudian *function* **proses_tambah_agenda()** digunakan untuk memproses masukan dari *form* dan menambahkannya ke tabel **agenda** melalui *model* **agenda_model**. Setelah memasukkan data ke tabel **agenda**, tampilan web dialihkan ke halaman **index** yang ada di *controller* **daftaragenda**.

Untuk *view* **form_tambah_agenda,** didalamnya terdapat sebuah *form* yang mengarahkan proses ke *function* **proses_tambah_agenda** di *controller* **daftaragenda,** kemudian terdapat sebuah *text area* yang akan menerima masukan judul agenda dan keterangan agenda. Simpanlah *view* dibawah ini dengan nama **form_tambah_agenda.php** dan simpan di **application -> views -> daftaragenda.** *Source code* **form_tambah_agenda** dapat Anda lihat sebagai berikut:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Daftar Hadir Praktikum</title>
  </head>
  <body>
    <h2>Form Tambah Agenda</h2>
    <fieldset>
      <form action="<?php echo
        site_url('daftaragenda/proses_tambah_agenda');?>" method="POST"> Nama
        : <br/><textarea name="nama" cols="50" rows="5"></textarea>
        <br/><br/>
        Keterangan : <br/><textarea name="keterangan" cols="50" rows="5"></textarea>
        <br/><br/>
        <input type="submit" value="Tambah" />
      </form>
    </fieldset>
  </body>
</html>
```

Mari kita coba hasil dari pekerjaan kita diatas. Pertama klik menu ***tambah agenda*** *di* halaman **index** di *controller* **daftaragenda**. Perhatikan gambar berikut:



Kemudian jika berhasil, Anda akan melihat tampilan berikut:


Daftar Hadir Praktikum - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Generating Query Results : CodeIgnit...
Daftar Hadir Praktikum
localhost/pelatihanbasdat/index.php/daftaragenda/tambah_agenda
Form Tambah Agenda
Nama :
Keterangan :
Tambah