

1. Berkenalan dengan CodeIgniter

1.1. Apa itu 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 penggunaannya akan menjelajahi *web* melalui halaman beranda. Sedangkan *framework* adalah desain struktur dasar yang dapat digunakan kembali (*reuseable*) 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 lebih cepat dibandingkan dengan menulis kode dari awal (*scratch*). CodeIgniter menyediakan kumpulan *library* untuk tugas – tugas yang sering dilakukan (*commonly needed task*) dan sangat mudah untuk mengakses *library* yang tersedia di CodeIgniter. Dengan menggunakan CodeIgniter, kita cukup fokus pada pengembangan proyek dan meminimalisir jumlah kode yang akan ditulis.

Sebagai *web framework* yang populer yang menggunakan bahasa pemrograman PHP, CodeIgniter mempunyai berbagai keunggulan seperti yang disebutkan di dokumentasinya:

1. *Free*, karena berada dibawah lisensi *open source* mirip Apache/BSD, kita dapat melakukan apapun dengan CodeIgniter. Lisensi lengkapnya dapat dilihat di halaman dokumentasi
2. *Light Weight*, sistem inti CodeIgniter memerlukan *library* yang sedikit. Berbeda sekali dengan *framework* lainnya yang membutuhkan banyak sumber daya tambahan. *Library* tambahan akan digunakan ketika request secara dinamis, membuat sistem yang dibangun menjadi efisien dan cukup cepat
3. *Fast*, menurut dokumentasi, performa yang dimiliki CodeIgniter terbukti cepat setelah dibandingkan dengan *web framework* lainnya
4. Menggunakan kaidah M-V-C, Dengan menggunakan Model-View-Controller, kita dapat memisahkan bagian logic dan presentation dari aplikasi yang kita bangun. Hal ini tentu sangat cocok dan bagus untuk proyek yang memfokuskan desainer fokus pada template file dan programmer fokus pada pembangunan logic dari aplikasi yang dibangun
5. Menghasilkan URL yang bersih, URL yang dihasilkan oleh CodeIgniter bersih dan ramah terhadap mesin pencari. CodeIgniter menggunakan pendekatan *segment-based* dibandingkan dengan *query string* yang biasa digunakan oleh *programmer* yang tidak menggunakan *web framework*. Berikut adalah contoh URL yang dihasilkan CodeIgniter:

cs.upi.edu/news/post/123

6. *Packs a Punch*, CodeIgniter hadir dengan berbagai *library* yang akan membantu tugas – tugas di pengembangan *web* yang sudah umum dan sering dilakukan seperti mengakses *database*, mengirim *email*, validasi data dari *form*, mengelola *session*, manipulasi gambar, bekerja dengan XML-RPC dan masih banyak lagi
7. *Extensible*, kita dapat menambahkan *library* atau *helper* yang kita ciptakan sendiri ke dalam CodeIgniter. Selain itu kita dapat juga menambahkan fitur lewat *class extension* atau *system hooks*.
8. *Thoroughly Documented*, hampir semua fitur, *library*, dan *helper* yang ada di CodeIgniter telah terdokumentasi dengan lengkap dan tersusun dengan baik. Ketika mendapatkan unduhan CodeIgniter, dokumentasinya sudah tersedia dan siap digunakan
9. Mempunyai komunitas yang ramah, komunitas CodeIgniter sangat ramah dan siap membantu pengguna CodeIgniter pemula atau yang sudah mahir. Komunitasnya dapat ditemui di : <http://codeigniter.com/forums/>

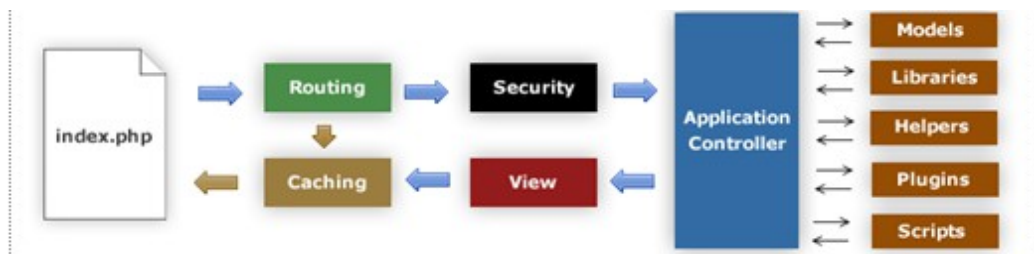
Untuk mendukung pengembangan aplikasi yang kokoh, Code Igniter memiliki fitur – fitur seperti berikut:

1. *Model-View-Controller Based System*
2. *PHP 4 Compatible*
3. Sangat ringan
4. Fitur lengkap untuk beberapa *engine database*.
5. *Active Record*
6. Form dan validasi data
7. Keamanan dan XSS *filtering*
8. *Session Management*
9. *Email Sending Class*. Supports Attachments, HTML/Text email, *multiple protocols* (sendmail, SMTP, dan Mail) dan lainnya
10. *Image Manipulation Library* (*cropping, resizing, rotating, dsb.*). Mendukung GD, ImageMagick, dan NetPBM
11. *File Uploading Class*
12. *FTP Class*
13. *Localization*
14. *Pagination*
15. *Data Encryption*
16. *Benchmarking*
17. *Full Page Caching*
18. *Error Logging*
19. *Application Profiling*
20. *Scaffolding*

21. *Calendaring Class*
22. *User Agent Class*
23. *Zip Encoding Class*
24. *Template Engine Class*
25. *Trackback Class*
26. *XML-RPC Library*
27. *Unit testing class*
28. *Search-engine friendly URLs*
29. URI routing yang fleksibel
30. Mendukung *hooks*, *class extension* dan *plugins*
31. Pustaka *helper* yang lengkap

1.2.Cara Kerja CodeIgniter

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



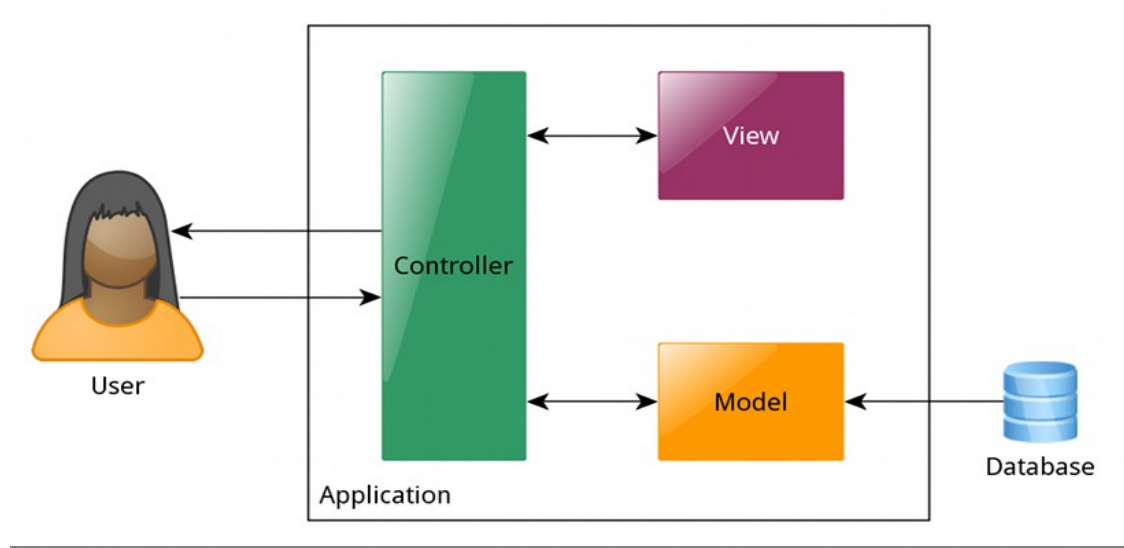
Gambar 1.1 Cara Kerja CodeIgniter

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*. Jika *caching* dijalankan, *view* akan di-cache terlebih dahulu agar dapat ditampilkan di *request* selanjutnya

1.3.Cara Kerja MVC



Gambar 1.2 Cara Kerja MVC (Sumber : <http://killer-web-development.com/section/1/3>)

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 lainnya yang dibutuhkan untuk menangani HTTP request dan menghasilkan halaman web.

Tapi pada CodeIgniter, *developer* juga dapat mengabaikan pemakaian Model dan cukup menggunakan Controller dan View.

1.4. Daftar Istilah di CodeIgniter

Sebelum menjelajah CodeIgniter lebih dalam, ada beberapa istilah yang akan selalu hadir selama pelatihan berlangsung. Berikut adalah daftar istilah yang akan hadir selama pelatihan:

1. *Model*, *class* PHP yang dirancang untuk bekerja dengan informasi dari *database*
2. *Controller*, inti aplikasi yang menentukan penanganan *HTTP request*
3. *View*, halaman *web* seperti *header*, *footer*, *sidebar* dan lainnya yang ditanamkan di halaman *web* yang lainnya. *View* tidak pernah dipanggil secara langsung. *View* harus dipanggil dari *controller*
4. *Library*, *class* yang berisi fungsi – fungsi untuk penyelesaian kasus tertentu
5. *Helper*, pembantu tugas untuk kategori tertentu yang terdiri dari kumpulan *function*
6. *Driver*, *library* khusus yang mempunyai *class* induk dan beberapa *class* turunan yang dapat digunakan untuk kasus tertentu

1.5. Peralatan yang Diperlukan

Untuk membangun aplikasi *web* yang menggunakan CodeIgniter tentunya kita membutuhkan bahasa pemrograman PHP4 atau PHP5. Setelah PHP4 atau PHP5 terpenuhi, kita dapat menggunakan berbagai alat yang sudah dikenal luas. Alat – alat tersebut terbagi kedalam beberapa kategori yang akan menangani perannya masing – masing. Berikut adalah alat – alat dasar yang dapat digunakan untuk membangun aplikasi *web* menggunakan CodeIgniter:

1. *Text Editor*, digunakan untuk menulis *source code* PHP dan lainnya serta untuk menyunting *file* konfigurasi. Sebagai contoh, Notepad++, Sublime Text, Geany, Kate, Komodo Edit, Aksi IDE, Netbeans
2. *Web Server*, agar aplikasi *web* yang dibangun dapat diakses oleh pengguna. Sebagai contoh Apache, Lighttpd, IIS, Nginx
3. *Database Management System*, menyimpan informasi yang dibutuhkan oleh aplikasi yang dibangun. Sebagai contoh MySQL, SQLite3, MS-SQL Server, Oracle, PostgreSQL
4. *Internet Browser*, digunakan untuk menampilkan aplikasi dan berinteraksi dengan antarmukanya. Sebagai contoh, Internet Explorer, Opera, Google Chrome, Mozilla Firefox
5. *Mail Server*, memberikan layanan e-mail kepada *user* dan mengintegrasikannya dengan aplikasi *web* yang akan dibangun. Sebagai contoh, SquirrelMail, ArgoSoft, RoundCube, GMail
6. *PDF Creator*, digunakan untuk menghasilkan *file* dalam bentuk PDF. Sebagai contoh, PDFLib, FPDF,
7. *Database Browser*, digunakan untuk melihat *database* secara grafikal. Sebagai contoh, PHPMyAdmin, MySQLBrowser, Chive, SQLite3 Manager, PgAdmin, SQL Server Browser, Oracle Apex
8. Sistem Operasi, landasan untuk mengembangkan aplikasi *web* sebelum diluncurkan di *server*. Sebagai contoh, Windows 7, Windows XP, Ubuntu, Fedora, Slackware, BlankOn,

IGOS Nusantara, FreeBSD

9. Dan berbagai alat lainnya yang dapat digunakan oleh PHP atau memiliki interoperabilitas terhadap PHP

Dalam praktiknya, terdapat juga paket – paket yang menyertakan *web server*, *database management system*, beserta *database browser* dalam satu paket. Seperti XAMPP, WAMP, atau LAMPP yang cukup sekali *install*, kita sudah dapat menggunakan *web server*, *database management system*, *database browser*, dan fungsional *server* lainnya.

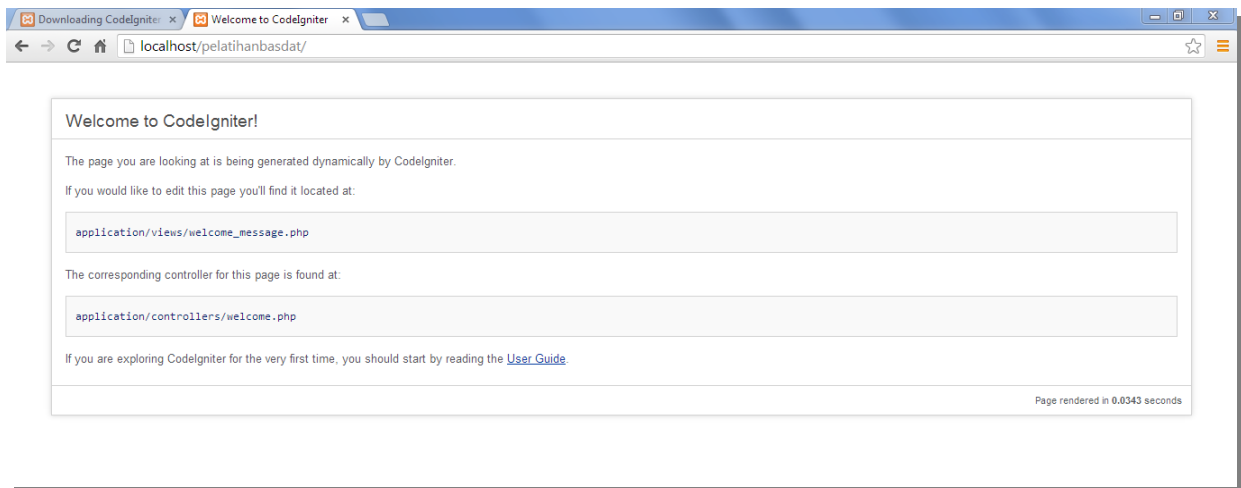
Untuk praktikum kita dari awal hingga akhir, akan digunakan CodeIgniter versi 2.1.4, paket XAMPP (terdiri dari Apache *Web Server*, MySQL *Database Management System*, dan PHPMyAdmin *Database Browser*), *internet browser* Firefox atau Google Chrome, *text editor* seperti Notepad++ atau Geany, dan sistem operasi Windows 7.

2. Memulai CodeIgniter

2.1. 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 : <http://ellislab.com/codeigniter/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. nyalakan Apache (*web server*) dan MySQL (*database management relationship*) melalui XAMPP Control Panel
6. akses direktori tersebut lewat *browser* dengan URL : <http://localhost/pelatihanbasdat>
7. Jika berhasil akan muncul tampilan seperti berikut:



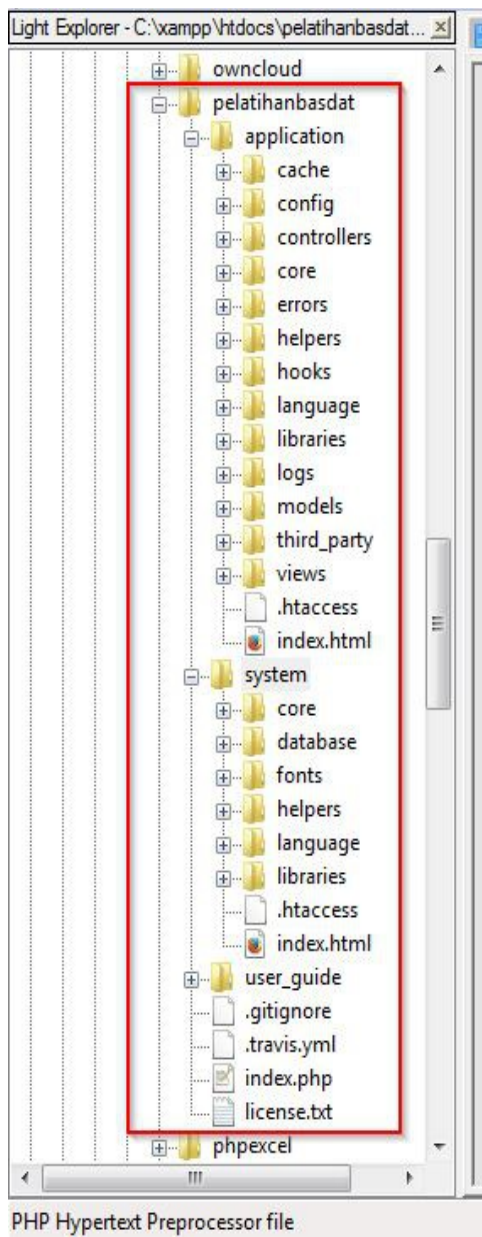
Gambar 2.1 CodeIgniter saat pertama kali di install

8. Selamat Anda berhasil melakukan instalasi CodeIgniter

2.2. Mengenal Struktur Direktori CodeIgniter

Di CodeIgniter terdapat hirarki yang dikepalai oleh tiga *folder* utama, yaitu : **application**, **system**, dan **user_guide**. *Folder application* adalah tempat dimana *programmer* aplikasi *web* yang menggunakan CodeIgniter, akan menyusun aplikasinya. Berikut adalah peranan *folder application* di CodeIgniter:

1. menentukan halaman *error*
2. membangun *controller*
3. membangun *model*
4. membangun *views*
5. konfigurasi aplikasi *web* yang dibangun
6. membangun *library* sendiri
7. membangun *helper* sendiri



Gambar 2.2 Struktur Direktori CodeIgniter

Folder system adalah tempat dimana *programmer* aplikasi *web* yang menggunakan CodeIgniter, akan menyusun aplikasinya. Berikut adalah peranan *folder system* di CodeIgniter:

1. menyimpan *library* inti CodeIgniter di *folder core*
2. menyimpan *library* dan *driver* untuk *database* di *folder database*
3. menyimpan *font* yang dapat digunakan oleh keseluruhan aplikasi *web* secara *default* di *folder font*
4. menyimpan kumpulan *helper* yang dapat digunakan untuk membantu menyelesaikan tugas – tugas tertentu di *folder helpers*
5. menyimpan fitur bahasa yang disimpan di *folder language*
6. menyimpan berbagai *library* lainnya yang digunakan untuk pengembangan aplikasi *web* di *folder libraries*

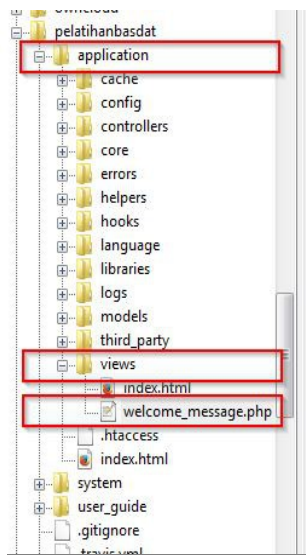
Folder user_guide adalah tempat dimana *programmer* aplikasi *web* yang menggunakan CodeIgniter, akan menyusun aplikasinya. Berikut adalah peranan *folder user_guide* di CodeIgniter:

1. Berisi *file* dokumentasi pengenalan CodeIgniter
2. Berisi *file* dokumentasi tentang pembahasan fitur – fitur umum yang dimiliki CodeIgniter
3. Berisi *file* dokumentasi tentang penggunaan *library* yang terdapat di CodeIgniter
4. Berisi *file* dokumentasi tentang penggunaan

helper yang terdapat di CodeIgniter

5. Berisi *file* dokumentasi tentang penggunaan *driver* yang terdapat di CodeIgniter

2.3. Mengubah Tampilan Awal CodeIgniter



Gambar 2.3 Mengubah Kode *welcome_message.php*

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>
```

Pada kode yang ditandai dengan warna kuning, kita ubah tulisan “Welcome to CodeIgniter!” menjadi “Selamat Datang di Pelatihan Basis Data 2014”, yang akhirnya *source code* diatas berubah menjadi seperti berikut ini:

```
<div id="container">
  <h1>Selamat Datang di Pelatihan Basis Data 2014</h1>

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

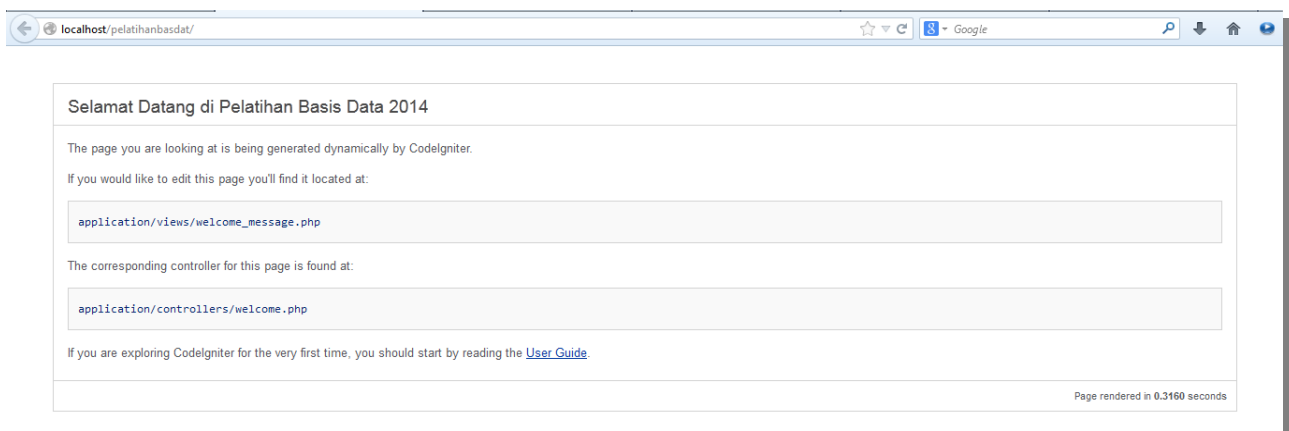
    <p>If you would like to edit this page you'll find it located at:</p>
    <code>application/views/welcome_message.php</code>

    <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>
```

Hasil dari pengubahan kode diatas akan menjadi seperti tampilan berikut ini:



Gambar 2.4 Halaman Welcome Message yang Berhasil Diubah

Bagaimanakah halaman diatas muncul ? Halaman diatas muncul karena kita memanggil sebuah *controller* yang bernama **welcome**. *Controller* ini terdapat di **pelatihan basdat -> application -> controllers -> welcome.php**. Di dalam *file* ini **welcome.php** terdapat *source code* seperti berikut:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Welcome extends CI_Controller {

    /**
     * Index Page for this controller.
     *
     * Maps to the following URL
     *      http://example.com/index.php/welcome
     *  - or -
     *      http://example.com/index.php/welcome/index
     *  - or -
     * Since this controller is set as the default controller in
     * config/routes.php, it's displayed at http://example.com/
     *
     * So any other public methods not prefixed with an underscore will
     * map to /index.php/welcome/<method_name>
     * @see http://codeigniter.com/user_guide/general/urls.html
    */
    public function index()
    {
        $this->load->view('welcome_message');
    }
}

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

Mari kita simak bagian kode yang ditandai oleh warna kuning diatas:

1. **class Welcome extends CI_Controller { .. }**, merupakan *controller* di CodeIgniter yang direpresentasikan dalam sebuah *class*. *Class* ini dibangun untuk mengelompokkan fungsional *web* dalam yang akan diakses melalui URL. Setiap *controller* memiliki **function** yang mewakili cabang dari *controller* sebagai URI utama. Perintah **extends** digunakan untuk memanggil semua fungsional *controller* CodeIgniter. Istilahnya adalah diwariskan atau diturunkan (*inherit*) dalam teori pemrograman berorientasi objek. Di CodeIgniter, dapat dibuat banyak *controller* tergantung kebutuhan sistem yang akan dibangun. Misal di dalam *controller welcome* terdapat sebuah **function** yang bernama **lihat_info()**. Maka ketika kita memanggil **lihat_info()** di *controller welcome* akan menjadi :

http://pelatihanbasdat/index.php/welcome/lihat_info

2. **public function index () { }**, merupakan cara untuk membuat sub URI dari sebuah *controller*. Jika sub URI tersebut bernama **index()** maka ketika sub URI tersebut diakses tidak perlu ditulis langsung seperti pada contoh sebelumnya di poin no 1. Jika mempunyai nama selain **index()** misalnya **lihat_info()** maka ketika memanggil sub URI tersebut, harus ditulis secara jelas. Di dalam sebuah **function** di *controller* Anda dapat menentukan logika dari aplikasi Anda. Sebagai contoh, Anda dapat menampilkan halaman tertentu, atau menampilkan jumlah mahasiswa yang mengontrak mata kuliah basis data
3. **\$this->load->view('welcome_message')**, perintah ini digunakan untuk menampilkan halaman *web* dari *view* tertentu. Parameter **welcome_message** merupakan *view* yang akan ditampilkan ke halaman *web* ketika *user* meminta *request* terhadap URL tersebut. *View* ini terdapat di **application -> views -> welcome_message.php**

2.4.Membuat *Function* Baru di *Controller*

Sekarang kita akan membuat *function* baru di *contoller* **welcome**. *Function* tersebut adalah **lihat_info** yang akan menampilkan informasi seputar pelatihan basis data. Dengan menggunakan *text editor* favorit Anda, sunting **welcome.php** di **application->controllers** kemudian tambahkan kode berikut setelah *function* **index()**. Berikut potongan kode yang harus ditambahkan:

```
.....
public function lihat_info()
{
    echo "<head><title>Pelatihan Code Igniter 2014</title></head>";
    echo "<h1>Pelatihan Code Igniter 2014</h1>";
    echo "<h3>Presented by Lab Basis Data Ilmu Komputer UPI</h3>";
    echo "CodeIgniter merupakan salah satu <i>web framework</i> PHP yang sangat mudah dan menarik.";
    echo "Saya menghadiri ini untuk mulai mengenal dan mempelajari CodeIgniter";
}
.....
```

Di dalam *function* **lihat_info()**, hanya terdapat kode yang mencetak dokumen HTML melalui perintah **echo** yang dimiliki oleh PHP. Setelah menambahkan kode diatas, akseslah *function* tersebut melalui URL berikut:

http://pelatihanbasdat/index.php/welcome/lihat_info

Jika berhasil, kode diatas akan menampilkan tampilan seperti berikut:



Gambar 2.5 Tampilan Lihat Info

2.5.Membuat *Function* Baru yang Menampilkan View Terpisah di *Controller*

Konsepnya hampir sama dengan subbab sebelumnya. Hanya saja kode *view* yang akan ditampilkan di *file* terpisah kemudian *file* tersebut dimuat di *controller*. Kita akan membuat sebuah *function* yang menampilkan biodata. Sebagai contoh berikut ini terdapat *file* yang berisi biodata fiktif seorang mahasiswa ilmu komputer. *File* tersebut berisi kode seperti berikut ini:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Biodataku</title>
</head>
<body>

<h1>Supeno Rapopo</h1>

<h3>Ilmu Komputer UPI 2010</h3>

<p>
    Kota : Bandung <br />
    Tanggal Lahir : 30 Februari 1990 <br />
    Hobi : Ngoding, Ngegenjreng Gitar, Ngegesek Biola <br />
    Tools Favorit : CodeIgniter, Django, LibGdx
</p>

</body>
</html>
```

Simpan *file* diatas di **application -> views** dengan nama **biodata.php**. Kemudian di *controller welcome* tambahkan potongan kode berikut untuk menampilkan *view* biodata yang telah ditulis. Berikut kode di *controller* untuk menampilkan *view* biodata:

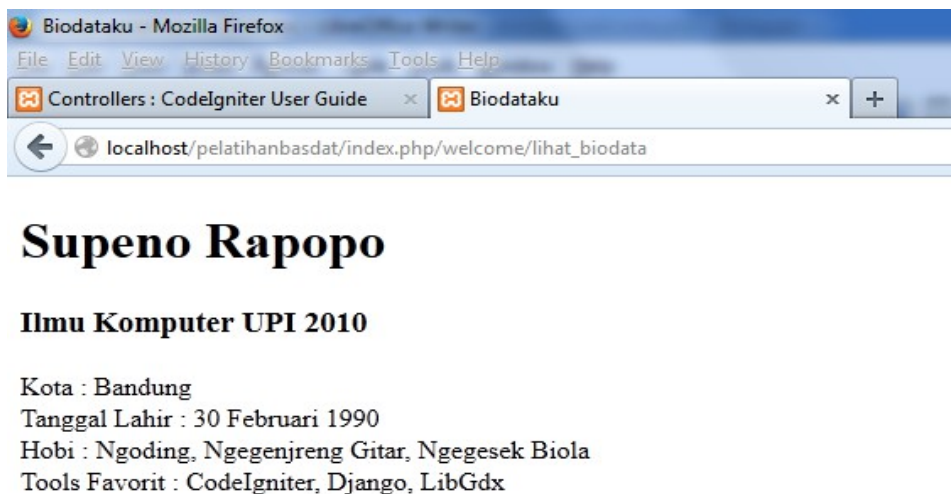


```
public function lihat_biodata()
{
    $this->load->view('biodata');
}
```

Di dalam *function* **lihat_biodata()**, hanya terdapat kode yang menampilkan *view* biodata dengan menggunakan perintah **load-view** yang tersedia di CodeIgniter. Setelah menambahkan kode diatas, akseslah *function* tersebut melalui URL berikut:

http://pelatihanbasdat/index.php/welcome/lihat_biodata

Jika berhasil, kode diatas akan menampilkan tampilan seperti berikut:



Gambar 2.6 Tampilan View Biodata

2.6. Menambahkan Argumen pada *Function* di *Controller*

Pada subbab sebelumnya, kita hanya membuat *function* yang tidak menggunakan argumen. dapat kita lihat pada **lihat_info()** tidak ada argumen yang dapat dilewatkan. Untuk itu kita akan mencoba membuat *function* yang mempunyai argumen. Diasumsikan kita akan membuat sebuah *function* yang bernama **jumlah_angka()**. *Function* ini mempunyai dua masukan yaitu \$angka1 dan \$angka2. Kemudian tempatkan *function* tersebut di *controller* **welcome**. Berikut adalah potongan

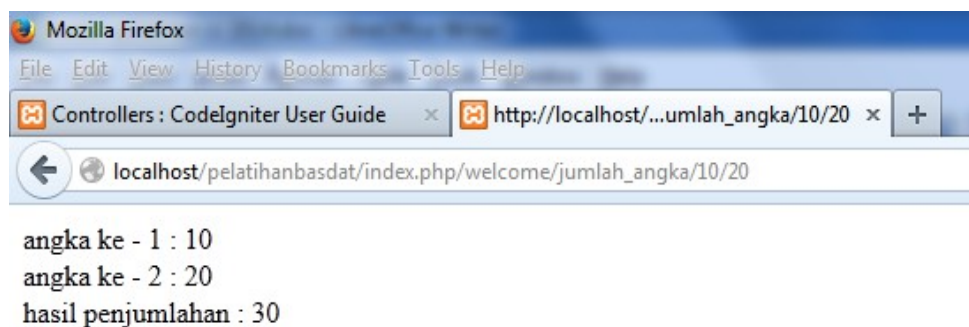
kode **jumlah_angka()**:

```
public function jumlah_angka($angka1, $angka2)
{
    if (isset($angka1) || isset($angka2)){
        echo "angka ke - 1 : $angka1 <br />";
        echo "angka ke - 2 : $angka2 <br />";
        $hasil = $angka1 + $angka2;
        echo "hasil penjumlahan : $hasil";
    }
    else {
        echo "Anda tidak memasukkan argumen kedalam function jumlah angka";
    }
}
```

Panggil *function* tersebut melalui URL berikut:

http://pelatihanbasdat/index.php/welcome/jumlah_angka/10/20

Jika berhasil akan tampil halaman seperti berikut ini:



Gambar 2.7 Tampilan View Jumlah Angka

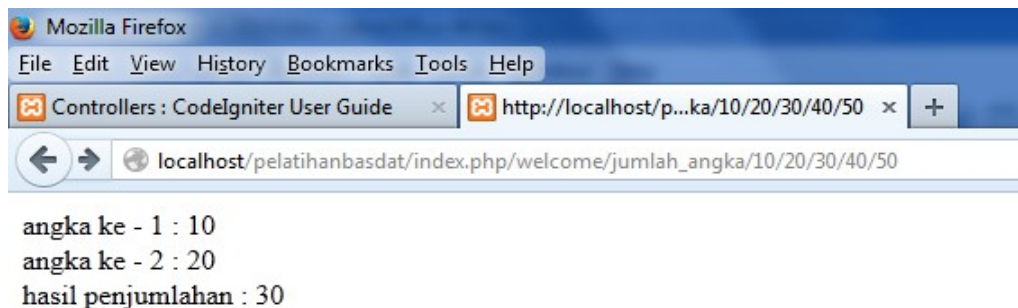
Bagaimanakah hal diatas bekerja ? Coba perhatikan URL diatas:

1. **welcome** adalah *controller* yang diakses oleh *user*
2. **jumlah_angka** adalah *function* yang terdapat di *controller welcome* yang memiliki

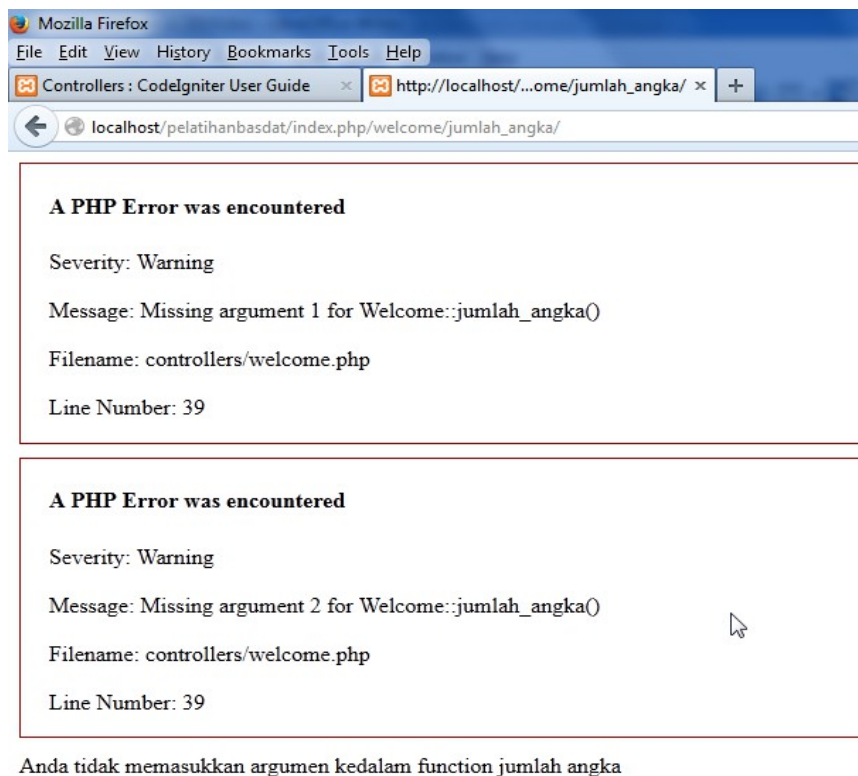
argumen \$angka1 dan \$angka2

3. 10 adalah nilai untuk \$angka1
4. 20 adalah nilai untuk \$angka2

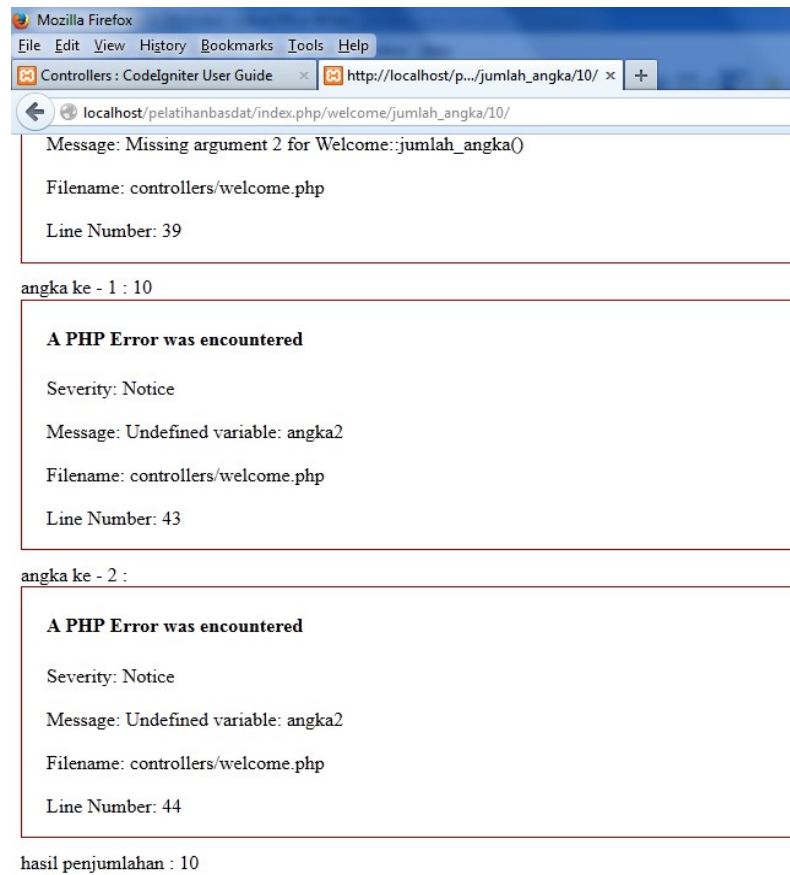
Menurut dokumentasi CodeIgniter, jika terdapat URI *segment* lebih dari dua. Maka URI *segment* yang ketiga dan seterusnya adalah argumen bagi *function* yang diakses oleh *user*. Dalam kasus ini adalah *function* **jumlah_angka()** menerima dua argumen yaitu 10 dan 20. Bagaimanakah jika argumen yang dilewatkan tidak sesuai ?. Berikut adalah beberapa contoh kasus jika argumen tidak dilewatkan sesuai dengan banyaknya argumen yang dibutuhkan:



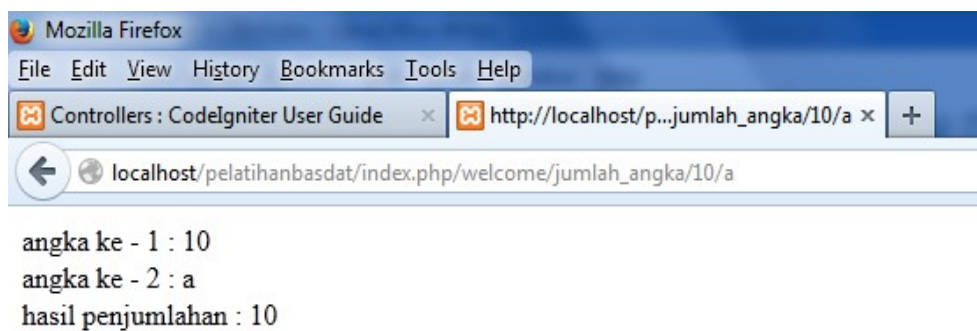
Gambar 2.8 Tampilan View Jumlah Angka Saat Kelebihan Argumen



Gambar 2.9 Tampilan View Jumlah Angka Saat Tidak Ada Argumen



Gambar 2.10 Tampilan View Jumlah Angka Saat Argumen yang Terpenuhi hanya Satu Argumen



Gambar 2.11 Tampilan View Jumlah Angka Saat Argumen yang Terpenuhi yang Salah Satunya bukan Angka

3. Mengenal *Controller* di CodeIgniter Lebih Jauh

3.1. 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:

Gambar 3.1 Tangkapan Layar Controller Calculator

```
<?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 */
```

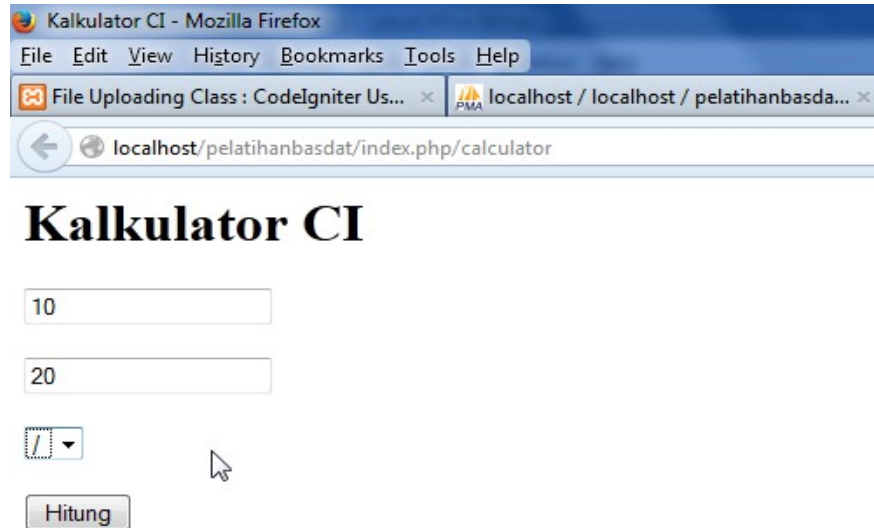
Sedangkan untuk *source code view* **hasil_hitung** adalah sebagai berikut:

```
<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**:



Kalkulator CI

10

20

/

Hitung

Gambar 3.2 Tangkapan Layar View Form Hitung Ketika Diberikan Masukan



Hasil perhitungan

10 / 20 = 0.5

[<< Kembali menghitung](#)

Gambar 3.3 Tangkapan Layar View Hasil Hitung

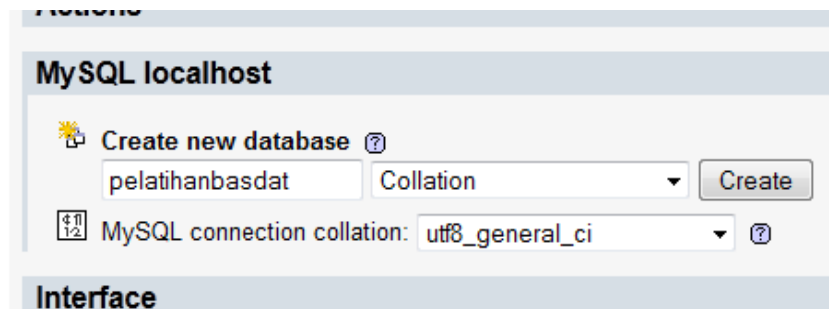
4. Mengenal Create Read Update Delete di CodeIgniter

4.1. 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*.

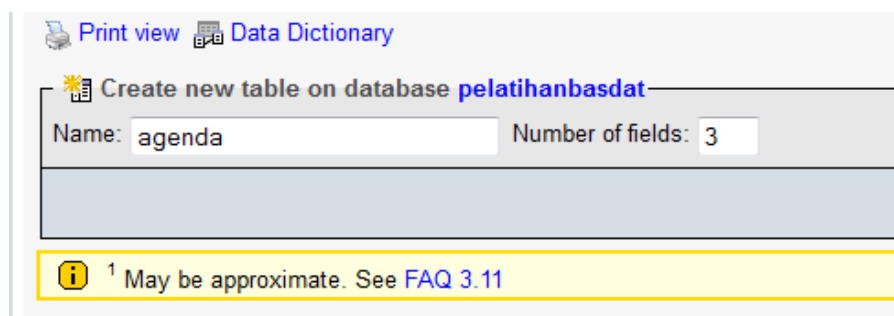
Dalam praktikum kali ini dibutuhkan MySQL sebagai sarana penyimpanan data yang akan digunakan oleh *user*. Kita akan membuat sebuah aplikasi yang mencatat agenda. Kita dapat mengisikan agenda baru, memperbaharui agenda yang sudah ada, melihat daftar agenda yang pernah dicatat, kemudian menghapus agenda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



Gambar 4.2 Membuat 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:

Field	id_agenda	nama	keterangan
Type	INT	VARCHAR	TEXT
Length/Values ¹		200	
Default ²	None	None	None
Collation			
Attributes			
Null	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index	PRIMARY	---	---
AUTO INCREMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comments			
MIME type			
Browser transformation			
Transformation options ³			

Table comments: Storage Engine: Collation:

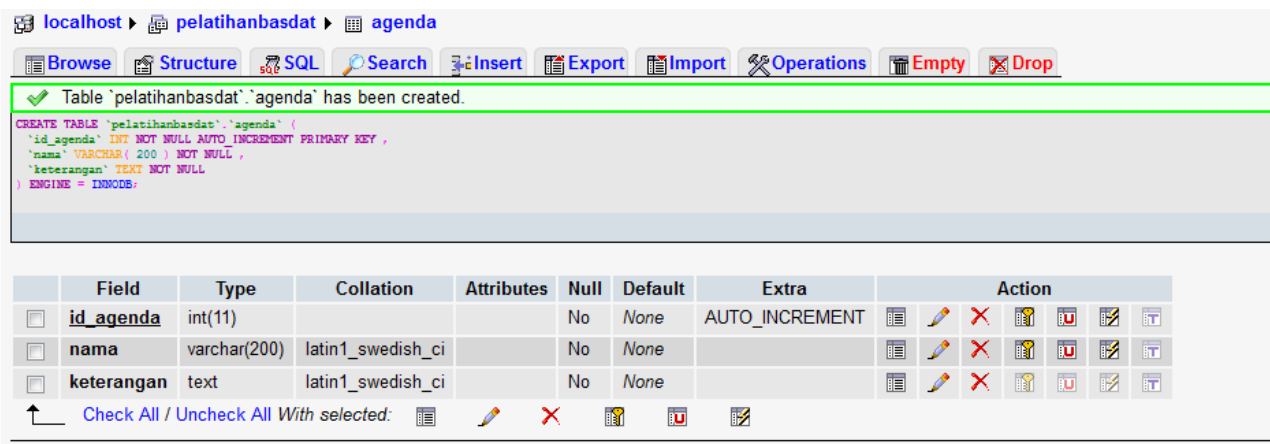
Gambar 4.3 Tangkapan Layar Ketika Mengisikan Field di Tabel Agenda

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



Gambar 4.4 Tangkapan Layar Tombol Save yang Harus Ditekan Ketika Selesai Mengisikan Field

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:



Gambar 4.5 Tangkapan Layar dari Tabel Agenda 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.

4.2. 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']['password'] = '';
$db['default']['database'] = 'pelatihanbasdat';
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;

```

```

$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;

/* End of file database.php */
/* Location: ./application/config/database.php */

```

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:
|
|         $autoload['libraries'] = array('database', 'session', 'xmlrpc');
*/
$autoload['libraries'] = array('database');
.....

```

Nah, seperti itulah cara konfigurasi *database* di CodeIgniter, sampai fase ini kita belum dapat menggunakan *database* untuk dipanggil di *controller*. Kita harus membuat *model* terlebih dahulu yang akan dipergunakan sebagai perantara ketika mengakses *database*.

4.3.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 menambahkan data baru.

Di CodeIgniter, kita dapat menggunakan sebuah cara yang dinamakan Active Record untuk mengakses *database*. Bentuknya adalah *function* yang siap pakai ketika CI_Model digunakan dalam *inheritance* kedalam *model* yang kita buat. Jadi tidak hanya menggunakan *query* yang biasa saja, dengan Active Record kita dapat membuat proses akses ke *database* lebih independen terhadap berbagai *database engine* dibandingkan *query* biasa. Karena setiap *query* biasa memiliki sintaks yang unik terhadap *database engine* yang menyediakan *query* tersebut. Singkatnya Active Record membuat aplikasi *web* Anda dapat beradaptasi dengan *database* manapun.

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*.

4.4.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 />

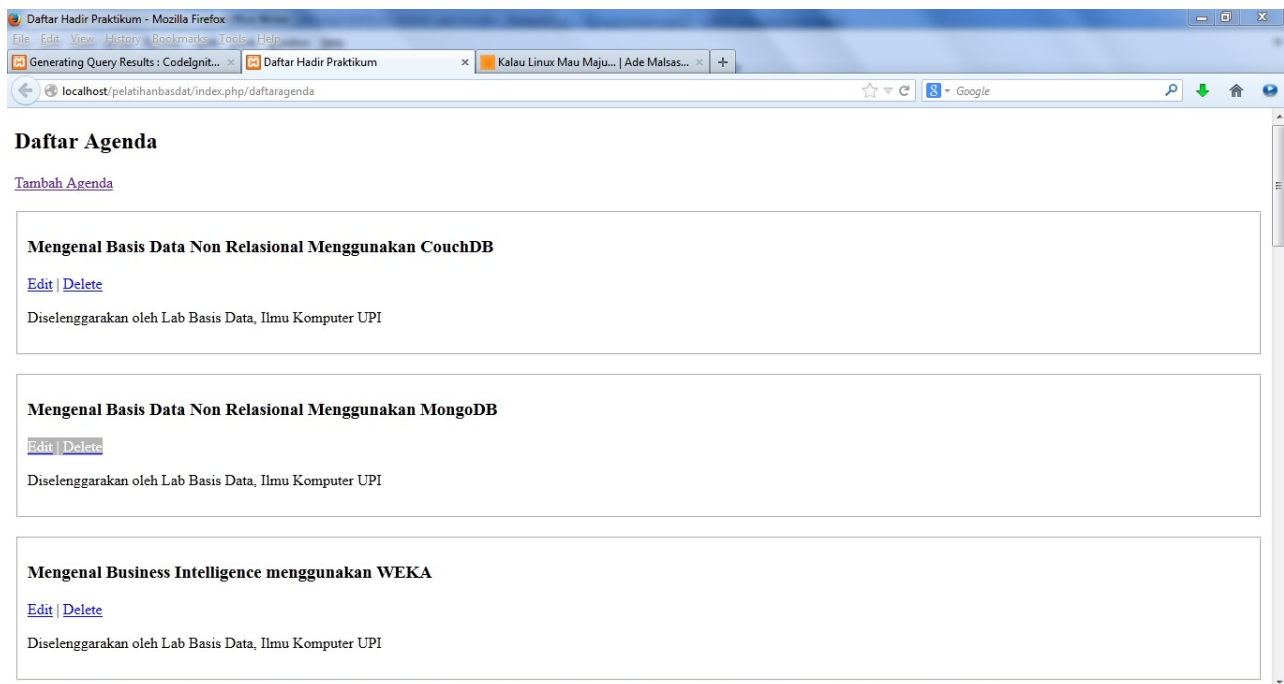
    <?php foreach ($daftar_agenda as $agenda) {?>
    <fieldset>
      <h3><?php echo $agenda->nama;?></h3>
      <a href="<?php echo site_url('daftaragenda/edit_agenda/'.$agenda->id_agenda);?>">Edit</a> |
      <a href="<?php echo site_url('daftaragenda/delete_agenda/'.$agenda->id_agenda);?>">Delete</a>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



Gambar 4.6 Tampilan Index di Controller Agenda

4.5. 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 CI_Controller {
```



```

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:

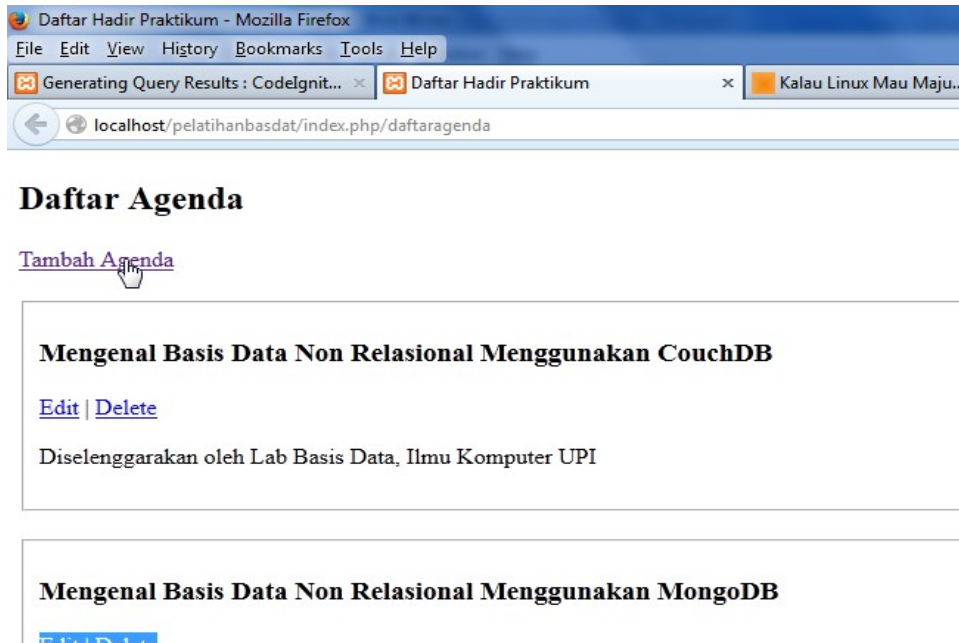
```

<!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');?&gt;" method="POST"&gt;
        Nama : &lt;br/&gt;&lt;textarea name="nama" cols="50" rows="5"&gt;&lt;/textarea&gt;
        &lt;br/&gt;&lt;br/&gt;
        Keterangan : &lt;br/&gt;&lt;textarea name="keterangan" cols="50" rows="5"&gt;&lt;/textarea&gt;
        &lt;br/&gt;&lt;br/&gt;
        &lt;input type="submit" value="Tambah" /&gt;
      &lt;/form&gt;
    &lt;/fieldset&gt;
  &lt;/body&gt;
&lt;/html&gt;
</pre

```

Mari kita coba hasil dari pekerjaan kita diatas. Pertama klik menu **tambah agenda** di

halaman **index** di *controller* **daftaragenda**. Perhatikan gambar berikut:



Gambar 4.7 Mencoba Menu Tambah Agenda

Kemudian jika berhasil, Anda akan melihat tampilan berikut:

Gambar 4.8 Melihat Tampilan Form Tambah Agenda

Sekarang kita coba masukkan sebuah agenda baru sebagai percobaan seperti pada gambar berikut

ini:

Daftar Hadir Praktikum - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Generating Query Results : CodeIgnit... x Daftar Hadir Praktikum x

localhost/pelatihanbasdat/index.php/daftaragenda/tambah_agenda

Form Tambah Agenda

Nama :

Mengenal AngularJS dalam 4 Jam

Keterangan :

Bersama Kang Toni Haryanto di Lab Praktikum, Ilmu KOmputer UPI

Tambah

Gambar 4.9 Menambahkan Agenda Baru

Setelah memberikan masukan, tentunya tekan tombol **Tambah** pada *form* tersebut. Lalu Lihatlah apa yang akan terjadi:

Generating Query Results : CodeIgnit... x Daftar Hadir Praktikum x

localhost/pelatihanbasdat/index.php/daftaragenda

Daftar Agenda

[Tambah Agenda](#)

Mengenal AngularJS dalam 4 Jam	Edit Delete
Bersama Kang Toni Haryanto di Lab Praktikum, Ilmu KOmputer UPI	

Gambar 4.10 Agenda Baru Berhasil Ditambahkan

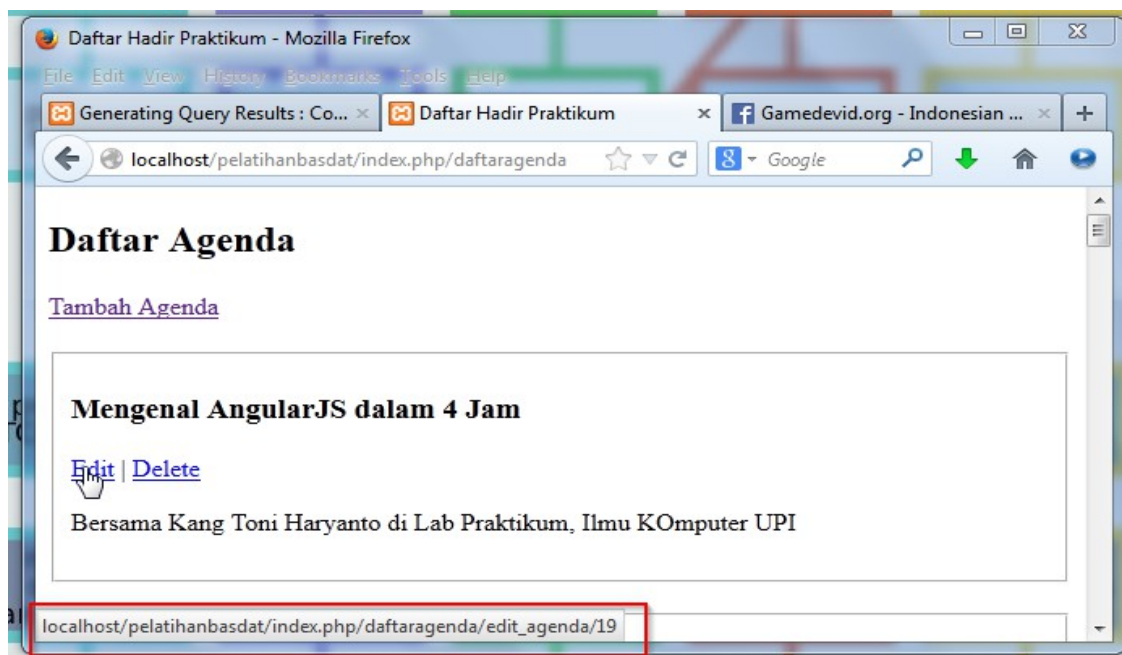
4.6. Menambahkan Fitur Edit Agenda di Controller Agenda

Sekarang kita beralih ke fitur selanjutnya di dalam CRUD yaitu *edit* agenda. Cara kerjanya hampir sama dengan menambahkan agenda baru. *Form* yang akan ditampilkan kepada *user* akan sama dengan *form* yang digunakan untuk tambah agenda. Hanya saja *form* tersebut sudah diisi data yang akan *edit*. Jadi *user* tidak perlu mengisi ulang data, cukup mengubah beberapa bagian data yang dirasa keliru.

Di setiap *link edit* yang terdapat di setiap *item*, terdapat **id_agenda** yang disematkan pada URL untuk mengarah ke halaman *edit*. Selengkapnya dapat dilihat pada potongan kode berikut (kode dibawah jangan disalin):

```
<?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 } ?>
```

Jadi ketika ditunjuk, URL akan seperti berikut ini:



Gambar 4.11 Mencoba Menu Edit di Daftar Agenda

Sebelum itu, kita harus membuat kode untuk menampilkan halaman *edit* agenda dan memproses hasil dari *edit* agenda tersebut. Tambahkan kode berikut di *controller agenda*:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Daftaragenda extends CI_Controller {

.....

    public function edit_agenda($id_agenda){
        $data['agenda'] = $this->agenda_model->select_by_id($id_agenda)->row();
        $this->load->view('daftaragenda/form_edit_agenda', $data);
    }

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

.....

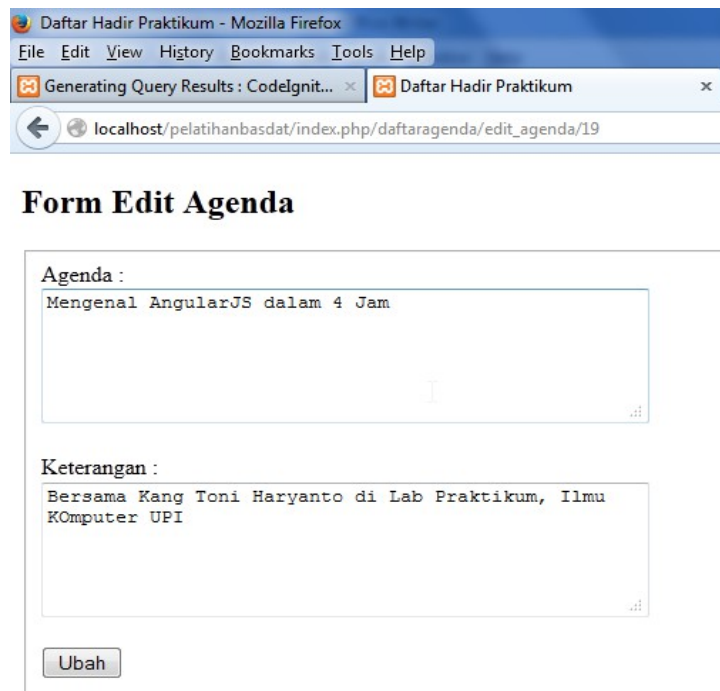
}
/* End of file welcome.php */
/* Location: ./application/controllers/welcome.php */
```

Kemudian agak berbeda sedikit dengan *view form_tambah_agenda*, isi *source code* dari *view form_edit_agenda* adalah sebagai berikut:

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

Simpan *source code* diatas dengan nama **form_edit_agenda.php** di **application -> views ->**

daftaragenda. Kemudian coba klik data yang baru saja dimasukkan atau data manapun untuk mencoba melihat halaman *edit* dari *item* tersebut:



Daftar Hadir Praktikum - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Generating Query Results : CodeIgnit... x Daftar Hadir Praktikum x

localhost/pelatihanbasdat/index.php/daftaragenda/edit_agenda/19

Form Edit Agenda

Agenda :

Mengenal AngularJS dalam 4 Jam

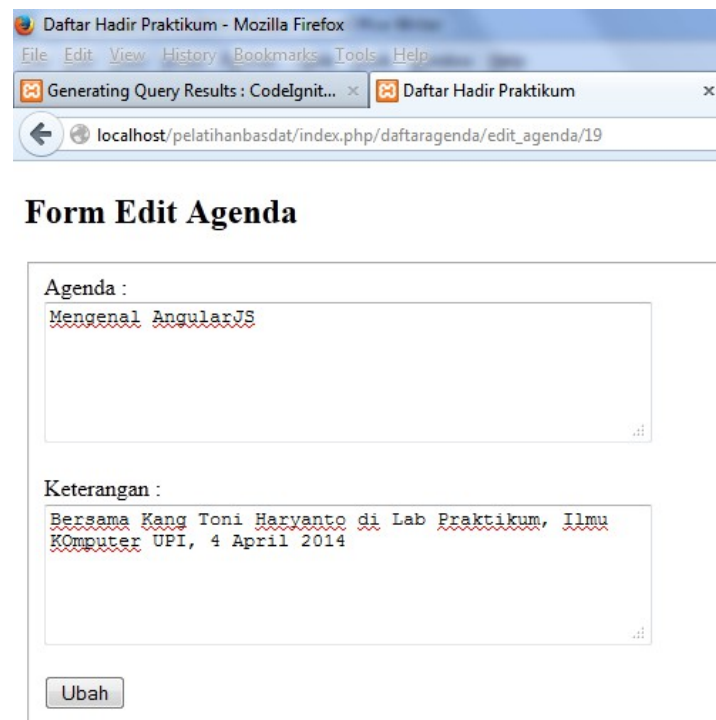
Keterangan :

Bersama Kang Toni Haryanto di Lab Praktikum, Ilmu KComputer UPI

Ubah

Gambar 4.12 Halaman Edit Agenda

Sekarang coba ubah sedikit data yang dipilih sesuai kehendak masing – masing. Misal dalam gambar dibawah ini diubah seperti berikut:



Daftar Hadir Praktikum - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Generating Query Results : CodeIgnit... x Daftar Hadir Praktikum x

localhost/pelatihanbasdat/index.php/daftaragenda/edit_agenda/19

Form Edit Agenda

Agenda :

Mengenal AngularJS

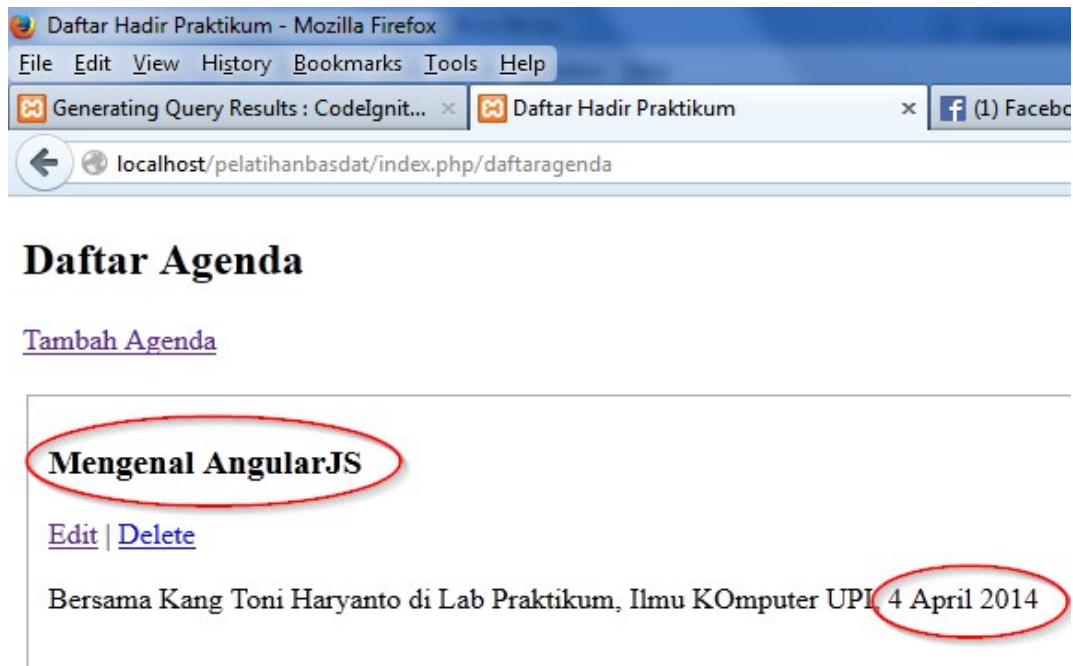
Keterangan :

Bersama Kang Toni Haryanto di Lab Praktikum, Ilmu KComputer UPI, 4 April 2014

Ubah

Gambar 4.13 Mencoba Edit Agenda

Hasil akhirnya adalah seperti berikut:



Gambar 4.14 Agenda yang Dipilih Berhasil Diubah

4.7. Menambahkan Fitur Hapus Agenda di Controller Agenda

Akhirnya kita mencapai fitur terakhir yaitu hapus agenda. Suatu saat, ada beberapa *item* yang dirasa keliru dan tidak diperkenankan masuk ke dalam *database*. Maka baiknya ada fitur hapus *item* tersebut agar tidak tercatat di *database*. Di aplikasi agenda ini, fitur hapus ditambahkan pada setiap *item* disamping menu *Edit*. Di dalam aplikasi ini hapus direpresentasikan dalam menu *Delete*. Di dalam menu tersebut terdapat **id_agenda** yang disisipkan untuk diproses di *function delete_agenda* di *controller daftaragenda*. Perhatikan potongan kode dibawah ini (keterangan : jangan disalin):

```
<?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 } ?>
```

Kemudian setelah mengklik *link* tersebut, proses selanjutnya akan berjalan di *function delete_agenda* di *controller agenda*. Berikut adalah potongan kode yang harus ditambahkan di *controller agenda*:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Daftaragenda extends CI_Controller {

.....

public function delete_agenda($id_agenda){
    $this->agenda_model->delete_agenda($id_agenda);
    redirect(site_url('daftaragenda'));
}

.....

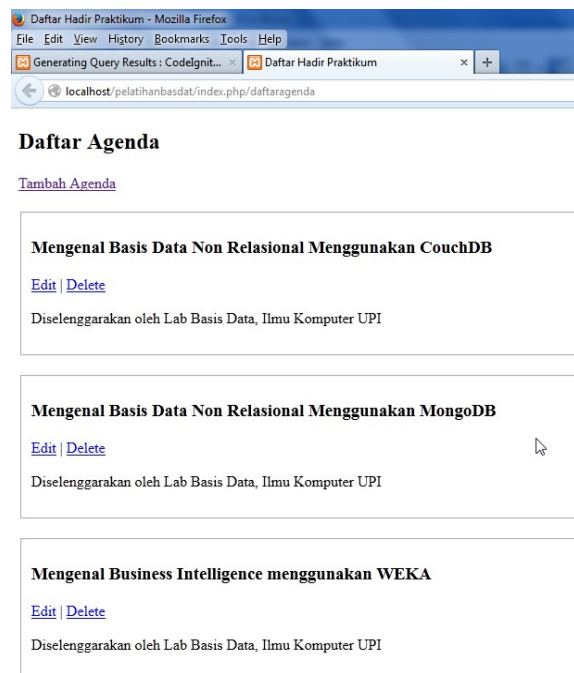
}

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

Di dalam *function* diatas, proses akan menghapus terlebih dahulu *item* yang berada di dalam tabel **agenda**. Kemudian akan dialihkan ke halaman daftar agenda. Berikut adalah tangkapan layar bagaimana proses hapus agenda berjalan:



Gambar 4.15 Salah Satu Agenda Sebelum Dihapus



Gambar 4.16 Setelah Proses Hapus Agenda Selesai

5.Implementasi Session CodeIgniter di Proses Autentikasi Akun

5.1.Membuat Tabel User di Database Pelatihan Basdat

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.

Agar percobaan sedikit lebih realistis, kita memerlukan sebuah tabel yang bernama **user**. Tabel ini mempunyai tiga buah *field* yaitu **id_user**, **username**, **password**. Dengan menggunakan PHPMyAdmin, 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		

Gambar 5.1 Struktur Tabel User di Database Pelatihan Basdat

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

	id_user	username	password
<input type="checkbox"/>	1	admin	admin
<input type="checkbox"/>	2	demo	demo

Check All / Uncheck All With selected:

Show: 30 row(s) starting from record 1

Gambar 5.2 Data Dummy untuk Tabel User

5.2.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();
    }
}
```

5.3.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 />
    Password : <input type="password" name="password" value="<?php echo set_value('password')?>" /> <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:

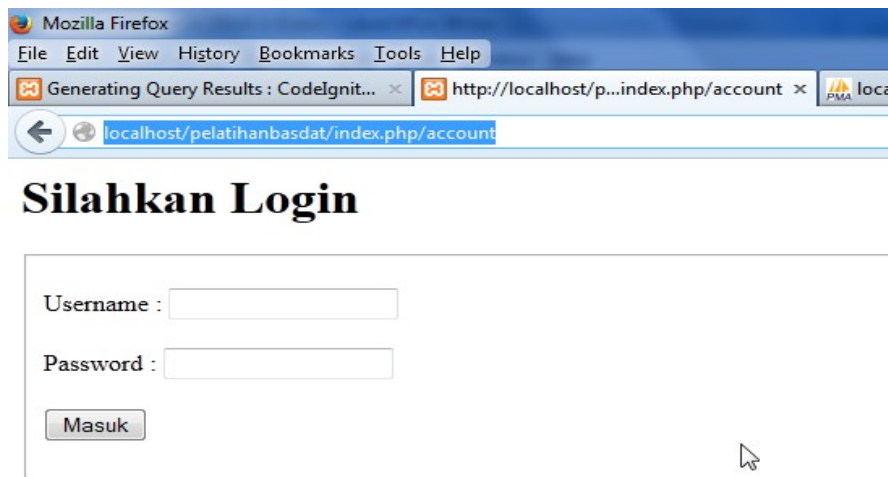
- **validation_error()**, akan menampilkan *error* yang dihasilkan ketika proses validasi *form*
- **\$this->session->flashdata('notification')**, akan menampilkan kesalahan ketika pengecekan keberadaan akun

- **set_value()**, mencetak kembali nilai masukan di *form* sebelumnya

Kemudian akses *function index* melalui URL berikut:

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

Maka akan muncul tampilan seperti berikut ini:



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 followed by a text box. Below these fields is a button labeled 'Masuk'. The browser's status bar at the bottom shows 'PMA local'.

Gambar 5.3 Tampilan Form Login

5.4. 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_succes_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');
    }

.....

}

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

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>
```


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



Gambar 5.4 Memasukkan Akun di Form Login

Kemudian jika berhasil, Anda akan diarahkan ke *function* **view_success_page()** seperti pada gambar berikut :



Gambar 5.5 Berhasil Ke Halaman Sukses

5.5. 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'));
    }

.....

}

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

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



Gambar 5.6 Mencoba Menu Logout

6. 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**. Berikut adalah *source code* lengkapnya:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Paginationssample 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,
        );

        // inialisasi 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 pagination sample
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 pagination sample*. Sekarang kita akan melihat *source code* dari *view* untuk menampilkan *pagination* yang bernama **daftar_agenda_pagination**. Berikut adalah *source code* dari *view 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 } ?>

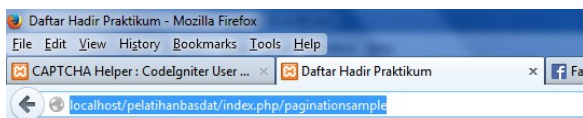
<br />
<?php echo $this->pagination->create_links(); ?>
<br />
<br />
</body>
</html>

```

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:



Daftar Agenda

[Tambah Agenda](#)

1 2 3 > Last >

Mengenal Basis Data Non Relasional Menggunakan CouchDB

[Edit](#) | [Delete](#)

Diselenggarakan oleh Lab Basis Data, Ilmu Komputer UPI

Mengenal Basis Data Non Relasional Menggunakan MongoDB

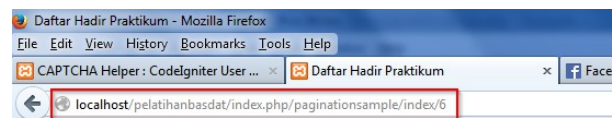
[Edit](#) | [Delete](#)

Diselenggarakan oleh Lab Basis Data, Ilmu Komputer UPI

Mengenal Business Intelligence menggunakan WEKA

[Edit](#) | [Delete](#)

Gambar 6.1 Tampilan Halaman Paging



Daftar Agenda

[Tambah Agenda](#)

< 1 2 3 4 5 >

Game Programming with Cocos2D

[Edit](#) | [Delete](#)

Diselenggarakan oleh Cengek GameDev, di Lab Praktikum Ilmu Komputer UPI

Pelatihan CCNA 2014

[Edit](#) | [Delete](#)

Diselenggarakan oleh Lab Jaringan Komputer, Ilmu Komputer UPI

Pelatihan PyGame 2014

[Edit](#) | [Delete](#)

Gambar 6.2 Halaman Pagination ketika Bernindah Halaman

7. Memahami Form Handling di CodeIgniter

7.1. 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 digunakan untuk eksperimen kita kali ini:

```
<?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 */
```

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:

Form Handling - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Form Handling x Welcome to CodeIgniter x Form Helper : CodeIgniter Us

localhost/pelatihanbasdat/index.php/formhandling/

Ayo Gabung Bersama Kami

Nama :

Samaran :

Email :

Password :

Ulang Password :

Umur :

Twitter :

Website :

Gambar 7.1 Form yang dibuat dengan Helper Form

```

1 <html>
2 <head>
3 <title>Form Handling</title>
4 </head>
5 <body>
6 <h1>Ayo Gabung Bersama Kami</h1>
7 <fieldset>
8
9 <form action="http://localhost/pelatihanbasdat/index.php/formhandling/proses_register_user" method="post" accept-charset="utf-8">
10 Samaran : <input type="text" name="samaran" value="" /> <br /><br />
11 Email : <input type="text" name="email" value="" /> <br /><br />
12 Password : <input type="password" name="password" value="" /> <br /><br />
13 Ulang Password : <input type="password" name="ulangpassword" value="" /> <br /><br />
14 Umur : <input type="text" name="umur" value="" /> <br /><br />
15 Twitter : <input type="text" name="twitter" value="" /> <br /><br />
16 Website : <input type="text" name="website" value="" /> <br /><br />
17 <input type="submit" name="daftar" value="Daftarkan Saya !" /> </form>
18 </fieldset>
19 </body>
20 </html>
21

```

Gambar 7.2 Source Code dari Form yang dibuat dengan Helper Form

7.2. 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 proses_register_user()* di *controller form_handling* yang akan kita gunakan untuk memvalidasi *form* yang telah dibuat:

```
<?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 */
```

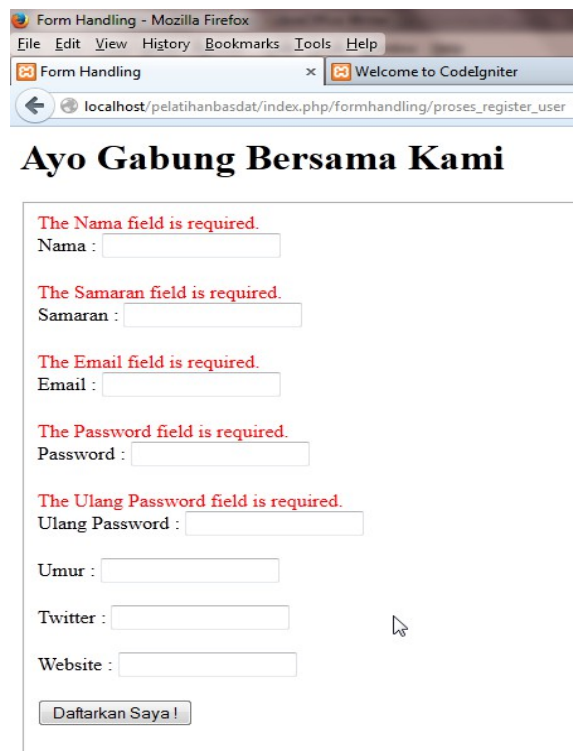
Bisa Anda lihat pada *source code* diatas bahwa untuk menentukan aturan validasi cukup dengan menggunakan **form_validation->set_rules()** dengan parameter pertama adalah nama dari *field* yang akan diberikan validasi, parameter kedua adalah nama dari *field* yang akan ditampilkan dalam peringatan *error*, 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 diisi



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 !

Gambar 7.3 Validasi Ketika Form Dikosongkan



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 !

Gambar 7.4 Validasi Ketika Salah Pengisian

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 !](#)

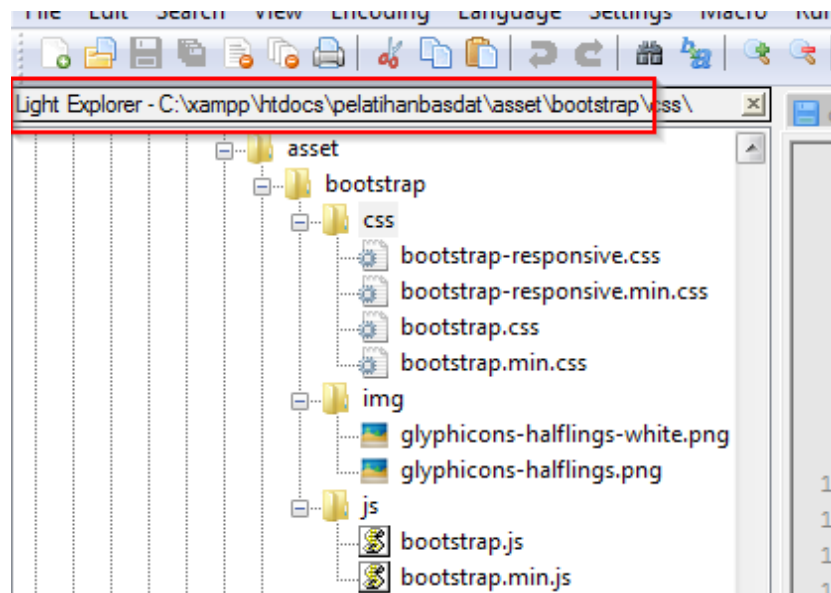
Gambar 7.6 Halaman Sukses Ketika Validasi berhasil

8. Menambahkan Twitter Bootstrap kedalam Proyek CodeIgniter

Tentunya dalam membangun sebuah aplikasi *web* diperlukanlah CSS untuk mempercantik tampilan *web*. Biasanya CSS dibangun sendiri atau menggunakan *framework* yang sudah menangani berbagai keperluan *styling* untuk sebuah aplikasi *web*. Dalam praktikum ini kita akan mencoba menggunakan Twitter Bootstrap dan memasangnya di proyek CodeIgniter.

Sudah pasti Anda membutuhkan Twitter Bootstrap untuk melakukan percobaan ini. Anda dapat mengunduhnya dari <http://getbootstrap.com/>. Kemudian unduh versi minimal atau versi yang disesuaikan dengan kebutuhan aplikasi *web* yang Anda bangun. Berikut ada beberapa langkah awal yang harus dilakukan untuk menggunakan Twitter Bootstrap di aplikasi *web* berbasis CodeIgniter:

- Di *root* dari *folder* proyek, buatlah sebuah *folder* yang bernama **asset**
- Kemudian ekstrak hasil unduhan Bootstrap dan ubah nama *folder*-nya menjadi **bootstrap**
- Sehingga hasilnya akan terlihat seperti pada struktur *folder* seperti ini:



Gambar 8.1 Struktur Folder Bootstrap yang Digabungkan dengan CodeIgniter

Seperti biasa kita membutuhkan *controller* yang akan menampilkan *view* yang berisi kode HTML. Dalam percobaan ini kita membutuhkan *controller* seperti berikut ini:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Bootstrapsample extends CI_Controller {

    public function __construct(){
        parent::__construct();
        $this->load->helper('url');
    }
}
```

```

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

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

```

Simpanlah *controller* diatas dengan nama **bootstrapsample.php** di **application -> controllers**. Berikutnya kita akan membuat *view* yang berisi contoh *form* yang menggunakan *style* dari Twitter Bootstrap. Mari kita simak *source code* berikut ini:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Contoh Form Bootstrap</title>
    <script type="text/javascript" src="<?php echo base_url('asset/bootstrap/js/bootstrap.js');?>"></script>
    <link href="<?php echo base_url('asset/bootstrap/css/bootstrap.min.css');?>" rel="stylesheet">
    <link href="<?php echo base_url('asset/bootstrap/css/bootstrap-theme.min.css');?>" rel="stylesheet">
</head>
<body>
    <form class="well form-horizontal">
        <fieldset>
            <div class="control-group">
                <label class="control-label" for="input01">Nama</label>
                <div class="controls">
                    <input type="text" class="input-xlarge" id="input01">
                    <p class="help-block">Isikan nama lengkap Anda</p>
                </div>
            </div>
            <div class="control-group">
                <label class="control-label" for="input02">Twitter</label>
                <div class="controls">
                    <input type="text" class="input-xlarge" id="input02">
                    <p class="help-block">Isikan akun Twitter Anda</p>
                </div>
            </div>
            <div class="control-group">
                <label class="control-label" for="select01">Pekerjaan</label>
                <div class="controls">
                    <select id="select01">
                        <option>Web Programmer</option>
                        <option>Desktop Programmer</option>
                        <option>Android Programmer</option>
                        <option>Web Designer</option>
                    </select>
                </div>
            </div>
            <div class="form-actions">
                <button class="btn btn-primary" type="submit">
                    Save changes
                </button>
                <button class="btn">

```

```

        Cancel
      </button>
    </div>
  </fieldset>
</form>
</body>
</html>

```

Simpanlah *source code* diatas dengan nama **contoh_form_bootstrap.php** kemudian simpan di *folder application -> views*. Sekarang kita akan meninjau beberapa bagian *source code* diatas. Mari kita simak:

- Untuk menggunakan Bootstrap kedalam halaman *view* kita maka dilakukan dengan cara seperti berikut (*source code* dibawah ini jangan disalin) :

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Contoh Form Bootstrap</title>
  <script type="text/javascript" src="<?php echo base_url('asset/bootstrap/js/bootstrap.js');?>"></script>
  <link href="<?php echo base_url('asset/bootstrap/css/bootstrap.min.css');?>" rel="stylesheet">
  <link href="<?php echo base_url('asset/bootstrap/css/bootstrap-theme.min.css');?>" rel="stylesheet">
</head>
<body>
.....

```

- Anda dapat menambahkan *class* **form-horizontal** pada *form* yang ditentukan untuk menciptakan *form* yang sesuai gaya Twitter Bootstrap
- *class* **control-group** digunakan untuk membuat sebuah susunan komponen *form* yang biasanya terdiri dari *label* dan *controls*
- *class* **control-label** digunakan untuk memberikan *style* bagi *label* dari *control* yang akan digunakan
- *class* **controls** merupakan sebuah susunan komponen *input* yang biasanya terdiri dari
- *class* **input-xlarge** memberikan *style* untuk *input* yang membuatnya terlihat lebih luas
- *class* **help-block** memberikan *style* untuk panduan ketika mengisi di *input*
- *class* **form-action** memberikan *style* untuk
- *class* **btn-primary** memberikan *style* untuk tombol yang ukurannya lebih besar

Untuk lebih lengkapnya Anda dapat membaca dokumentasi Twitter Bootstrap bagian **Base CSS**. Sekarang mari kita lihat tampilan dari *form* yang menggunakan Twitter Bootstrap melalui URL berikut ini:

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

Sedangkan tampilannya dapat dilihat seperti berikut ini:

Contoh Form Bootstrap - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Contoh Form Bootstrap

localhost/pelatihanbasdat/index.php/bootstrapsample

Nama

Isikan nama lengkap Anda

Twitter

Isikan akun Twitter Anda

Pekerjaan

Save changes Cancel

Gambar 8.2 Contoh Form Tampilan dengan Bootstrap

Contoh Form Bootstrap - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Contoh Form Bootstrap

localhost/pelatihanbasdat/index.php/bootstrapsample

Nama

Isikan nama lengkap Anda

Twitter

Isikan akun Twitter Anda

Pekerjaan

Web Programmer

Save changes Cancel

Gambar 8.3 Contoh Form yang Gagal Memuat Bootstrap

9.Membuat Fitur Upload di CodeIgniter

9.1.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>

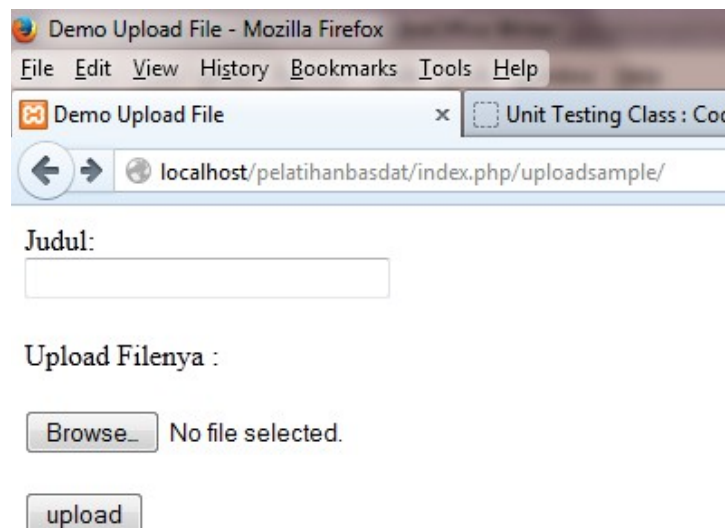
</body>
</html>

```

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/>



Gambar 9.1 Tangkapan Layar Form Upload

9.2. 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* proyek 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);
    }
}
}
/* End of file uploadsample.php */
/* Location: ./application/controllers/uploadsample.php */

```

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 mendapatkann 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*:

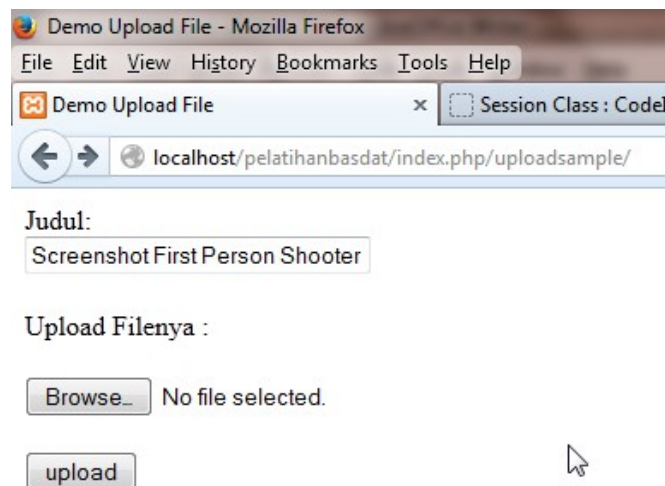
```
<html>
<head>
  <title>Demo Upload File</title>
</head>
<body>
  <h3>Proses Upload Berhasil !!!</h3>

  <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**.

Mari kita coba fitur *upload* ini dengan meng-*upload* foto yang Anda miliki. Mari kita lihat prosesnya:



Demo Upload File - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Demo Upload File x Session Class : Code

localhost/pelatihanbasdat/index.php/uploadsample/

Judul:

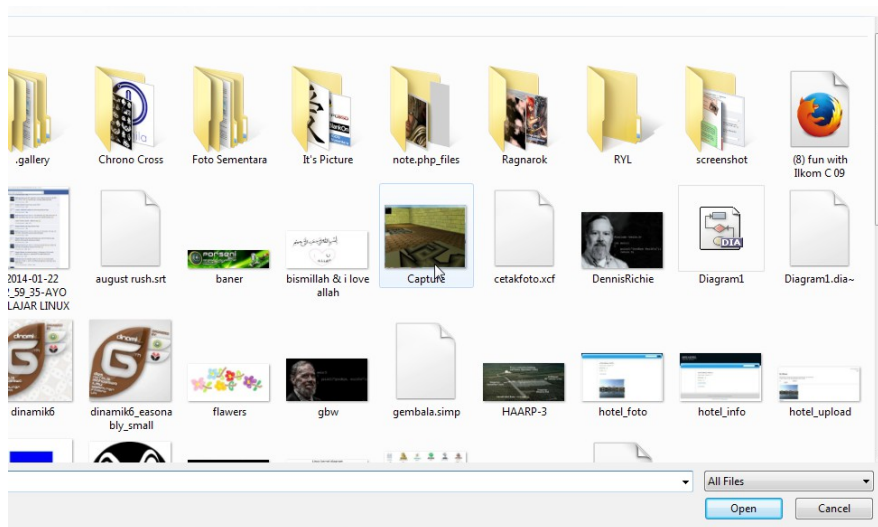
Screenshot First Person Shooter

Upload Filenya :

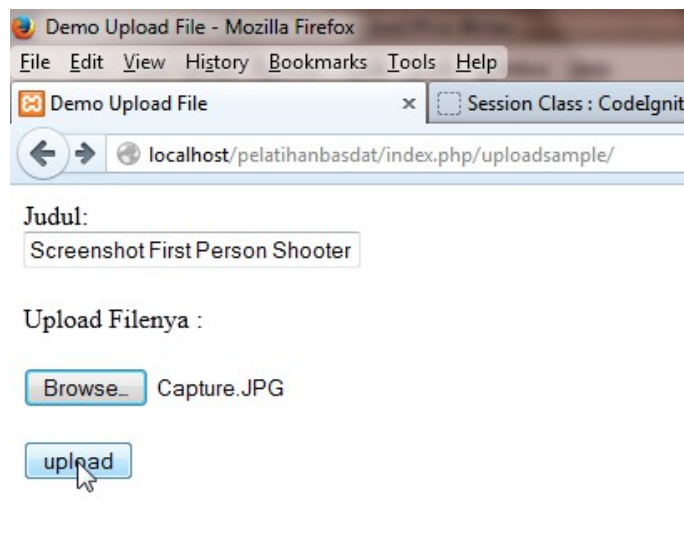
Browse_ No file selected.

upload

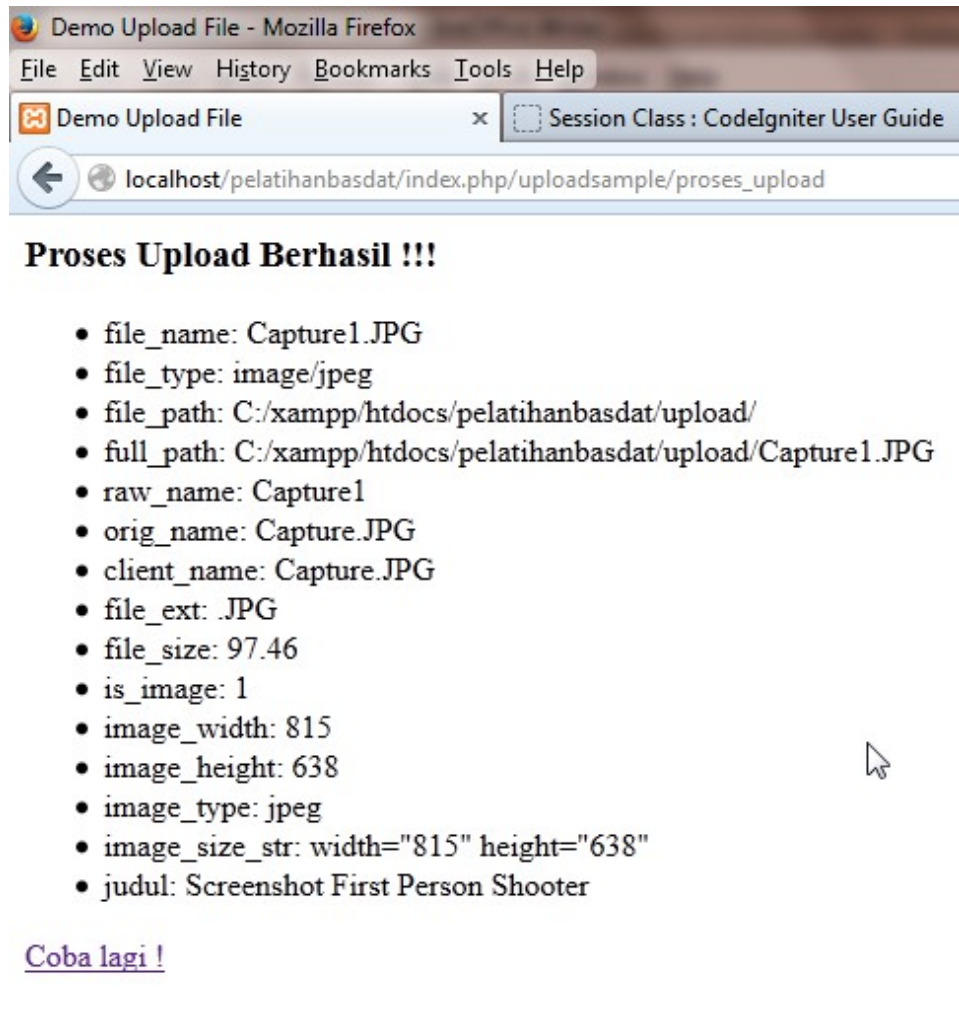
Gambar 9.2 Mengisi Judul pada Form Upload



Gambar 9.3 Memilih Gambar yang Akan Di Upload



Gambar 9.4 Menekan Tombol Upload



Gambar 9.5 Data – Data yang Dari File yang Di Upload

10. Menambahkan AJAX

10.1. Persiapan Menggunakan AJAX

AJAX atau *Asynchronous Javascript and XML* merupakan fitur yang terdapat di standard pembangunan sebuah *world wide web* (WWW). Dengan menggunakan AJAX, Anda tidak perlu memuat seluruh bagian halaman, cukup bagian tertentu saja yang dimuat ulang sesuai kebutuhan. Dengan demikian AJAX dapat menghemat beban pemuatan halaman *web* di sisi *client*.

Salah satu kekurangan AJAX adalah *history* yang tidak dapat dicatat ketika mengunjungi bagian – bagian *web*. Tidak seperti *web* tanpa AJAX yang dapat menyimpan *history* Anda ketika mengunjungi halaman *web* tertentu. Dan di percobaan kita kali ini akan digunakan AJAX yang terdapat di JQuery. Hal tersebut dipilih karena dengan JQuery, dapat meringkas waktu pengembangan aplikasi *web* yang menggunakan AJAX.

Tentunya agar percobaan AJAX ini lebih terlihat, maka diperlukanlah sebuah data yang disimpan di *database* untuk menampilkan data – data di aplikasi *web* ber-AJAX. Kita akan menggunakan tabel **agenda** yang terdapat di *database* **pelatihanbasdat**. Kita akan mencoba membuat fitur lihat agenda, tambah agenda, dan cari agenda. Dalam percobaan ini kita akan menggunakan *model* **agenda_model** yang sudah pernah digunakan sebelumnya. Hanya saja kita akan menambahkan *function* yang digunakan untuk melakukan pencarian agenda. Berikut adalah *source code* yang digunakan untuk melakukan pencarian di tabel **agenda**:

```
<?php
class Agenda_model extends CI_Model {
.....

// function yang digunakan oleh ajaxsample : proses_cari_agenda
function select_by_nama($nama){
    $this->db->select('*');
    $this->db->from('agenda');
    $this->db->like('nama', $nama, 'both');

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

Kemudian tentu kita memerlukan JQuery itu sendiri. Dalam percobaan kita kali ini diperlukan jquery-1.8.3.min.js. Anda dapat men-download-nya di <http://jquery.org>. Kemudian buatlah *folder* dengan nama **js** di **pelatihanbasdat** -> **asset**. Dan taruhlah JQuery di dalam *folder* **js**.

10.2.Membuat Fitur Melihat Agenda

Hal pertama yang akan kita pelajari adalah menampilkan isi tabel **agenda**. Jika pada bab 4 kita hanya menampilkan semua data melalui halaman utama. Kini kita akan memisahkannya pada *view* lain dan memuatnya kedalam bagian halaman. Hal ini dilakukan agar nanti ketika proses pencarian dapat menampilkan jumlah data secara dinamis.

Kita akan membuat *controller* dengan nama **ajaxsample** yang membutuhkan *model* **agenda_model**, *helper* **url**, dan *library* **input**. Kemudian kita akan menampilkan halaman utama yang terdapat di **view_daftar_agenda** dan menampilkan daftar agenda yang ada di **view_list_agenda**. Kenapa dipisah ? Karena dengan cara tersebut kita akan membuat bagian daftar agenda saja yang dinamis dan menanggapi perubahan. Sehingga tidak perlu semua halaman yang dimuat ulang ketika terjadi perubahan.

Simpan *source code* dibawah ini dengan nama **ajaxsample.php** di **application** -> **controllers**. Berikut adalah *source code* dari *controller* yang akan kita gunakan untuk melihat daftar agenda :

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Ajaxsample extends CI_Controller {

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

    // melihat halaman utama
    public function index(){
        $this->load->view('ajaxsample/view_daftar_agenda_ajax');
    }

    // proses untuk melihat detail agenda
    public function lihat_agenda(){
        $data['daftar_agenda'] = $this->agenda_model->select_all()->result();
        $this->load->view('ajaxsample/view_list_agenda', $data);
    }

}

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

Kemudian kita harus membuat halaman utama untuk menampilkan menu dan daftar agenda. Berikut adalah *source code* dari halaman utama yang akan kita bangun:

```
<!DOCTYPE html>
```

```

<html>
<head>
  <title>Daftar Hadir Praktikum</title>
  <script type="text/javascript" src="<?php echo base_url('asset/js/jquery-1.8.3.min.js');?>"></script>
</head>
<body>
  <h2>Daftar Agenda</h2>
  <a id="show-list-agenda" href="javascript:void(0)">Lihat Semua</a> |
  <a id="show-form-tambah-agenda" href="javascript:void(0)">Tambah Agenda</a> |
  <a id="show-form-cari-agenda" href="javascript:void(0)">Cari Agenda</a>
  <br /><br />
  <div id="view-form">
  </div>
  <br>
  <div id="view-list-agenda">
  </div>
</body>
<script type="text/javascript">
$(document).ready(function(){
  // menampilkan semua list agenda saat pertama kali halaman utama dimload
  $("#view-list-agenda").load('<?php echo site_url('ajaxsample/lihat_agenda');?>');

  // menampilkan semua list ketika menu lihat agenda ditekan
  $("#show-list-agenda").click(function(){
    $("#view-form").empty();
    $("#view-list-agenda").load('<?php echo site_url('ajaxsample/lihat_agenda');?>');
  });

  // menampilkan form tambah agenda
  $("#show-form-tambah-agenda").click(function(){
    $("#view-form").load('<?php echo site_url('ajaxsample/tambah_agenda');?>');
  });

  // menampilkan form cari agenda
  $("#show-form-cari-agenda").click(function(){
    $("#view-form").load('<?php echo site_url('ajaxsample/cari_agenda');?>');
  });

});
</script>
</html>

```

Sebelum menyimpan *source code* diatas, buatlah terlebih dahulu *folder* yang bernama **ajaxsample** di **application -> views**. Kemudian Simpanlah *source code* diatas dengan nama **view_daftar_agenda.php** di *folder ajaxsample*.

Mari kita simak beberapa bagian *source code* diatas:

- Bagian berikut merupakan cara untuk me-load JQuery ke dalam halaman *web*:

```
<script type="text/javascript" src="<?php echo base_url('asset/js/jquery-1.8.3.min.js');?>"></script>
```

- Bagian berikut akan menampilkan *form* di halaman utama:

```
<div id="view-form">
```

```
</div>
```

- Bagian berikut akan menampilkan daftar agenda:

```
<div id="view-list-agenda">
</div>
```

- **\$(document).ready(function());** digunakan untuk memulai membuka *script* JQuery
- **\$("#view-list-agenda").load('<?php echo site_url('ajaxsample/lihat_agenda');?>');** digunakan untuk menambatkan tampilan **lihat_agenda** dari *controller ajaxsample* ke dalam komponen yang memiliki id **view-list-agenda**
- **\$("#show-list-agenda").click(function());** digunakan untuk menangani *event* klik pada komponen yang memiliki id **show-list-agenda**

Dan terakhir kita harus membuat *view* yang berisi daftar agenda yang akan ditambatkan di halaman utama. Berikut adalah *source code* nya:

```
<?php foreach ($daftar_agenda as $agenda) {?>
    <fieldset>
        <h3><?php echo $agenda->nama;?></h3>
        <br /><br />
        <?php echo $agenda->keterangan;?>
    </fieldset>
    <br />
<?php } ?>
```

Simpan *source code* diatas dengan nama **view-list-agenda.php** kemudian simpan di **application -> views -> ajaxsample**. Sekarang kita coba akses *controller ajaxsample* melalui URL berikut:

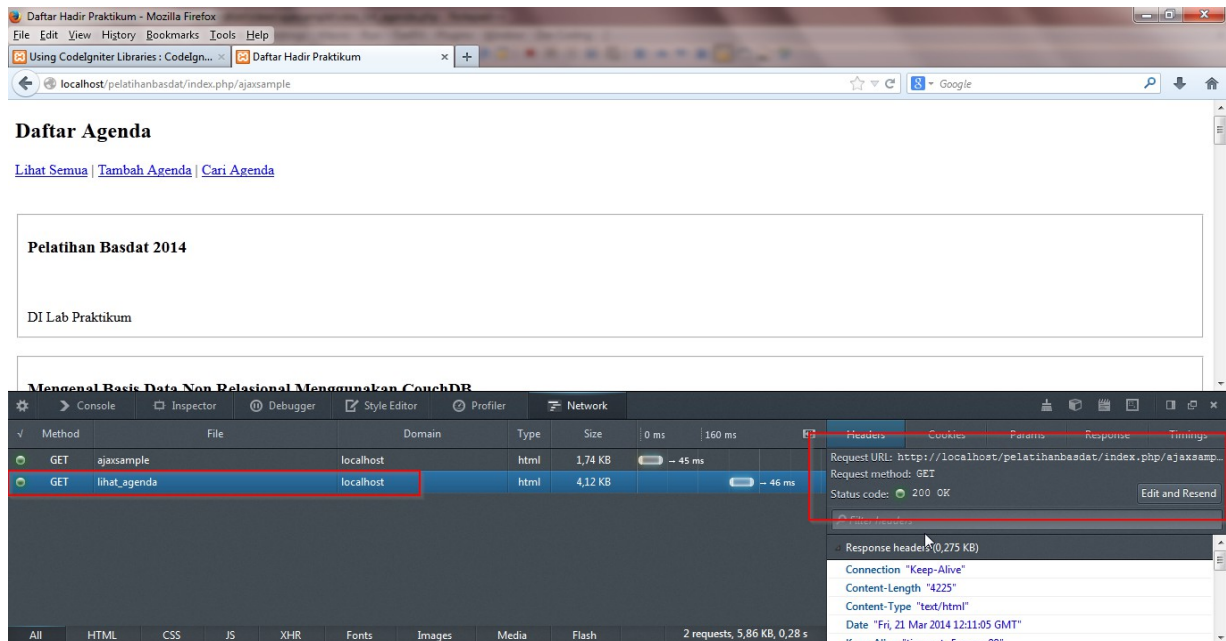
```
http://localhost/pelatihanbasdat/index.php/ajaxsample
```

Sebelumnya kita memerlukan *developer tools* tekan **Ctrl + Shift + K** jika Anda menggunakan Firefox atau tekan **Ctrl + Shift + J** jika Anda Menggunakan Chrome. Maka Anda akan melihat tampilan seperti berikut ini:

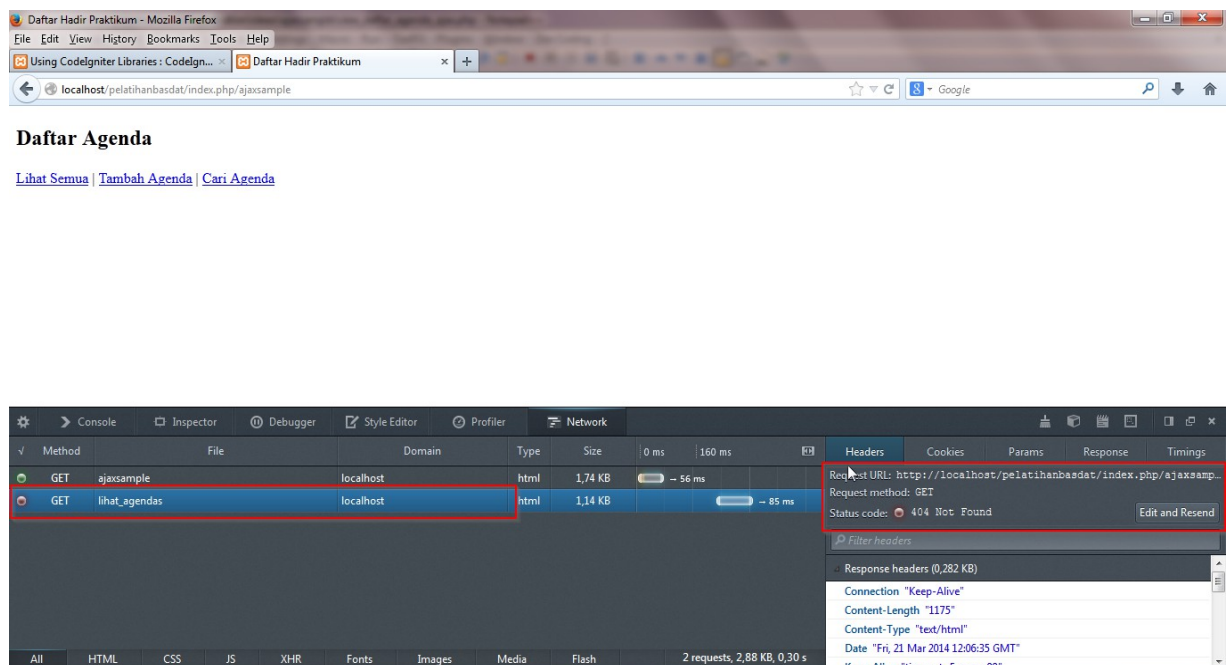


Ga
mbar 10.1 Halaman Web yang Mengandung AJAX

Melihat lebih dalam apa yang terjadi ketika AJAX dieksekusi di halaman *web* yang menggunakan AJAX:



Gambar 10.2 Halaman yang Berhasil Dimuat dengan AJAX Ketika Diperiksa Menggunakan Developer Tools pada Browser



Gambar 10.3 Halaman yang Gagal Dimuat dengan AJAX Ketika Diperiksa Menggunakan Developer Tools pada Browser

10.3. Menambahkan Cari Agenda

Selanjutnya kita akan mencoba menambahkan fitur *searching* yang menggunakan AJAX untuk mempercepat proses pencarian data. Tidak seperti *searching* yang menggunakan AJAX, *searching* seperti ini membuat Anda mendapatkan data lebih cepat dan interaktif ketimbang cara biasa.

Kita akan membuat sebuah *form* yang akan menampilkan hasil pencarian dengan AJAX. Ketika *user* mengetikkan kata kunci di *field* pencarian maka saat itu juga hasil akan dimunculkan melalui AJAX. Disini kita akan menggunakan *event keyup()* pada JQuery untuk melakukan hal tersebut.

Tentunya kita memerlukan sebuah *query* untuk mengambil data berdasarkan kemiripan kata kunci yang dituju. Dalam hal ini adalah judul agenda. *Query* tersebut sudah dibuat sejak awal di bab ini. Sekarang, tinggal menggunakan *query* tersebut di *controller ajaxsample*. Berikut adalah *function cari()* dan *proses_cari_agenda()* di *controller ajaxsample* yang akan digunakan untuk proses pencarian:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Ajaxsample extends CI_Controller {

.....

    // proses untuk mencari agenda
    public function cari_agenda(){
        $this->load->view('ajaxsample/form_cari_agenda');
    }

    public function proses_cari_agenda(){
        $nama = $this->input->post('nama');
        $data['daftar_agenda'] = $this->agenda_model->select_by_nama($nama)->result();
        $this->load->view('ajaxsample/view_list_agenda', $data);
    }

.....
}

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

Setelah membuat *function* yang dibutuhkan, kini kita akan membuat *form* pencarian yang terdiri dari *input* teks dan tombol batal. *Input* teks ini digunakan untuk menampung kata kunci, sedangkan tombol batal digunakan untuk menutup *form* jika ingin membatalkan pencarian.

Kemudian tombol batal akan dihubungkan dengan *event click()* yang akan melakukan penutupan *form*. Hal ini ditangani oleh JQuery. Sedangkan *input* teks dihubungkan dengan *event keyup()* dan *event* tersebut akan menangani proses penampilan hasil data yang hasilnya akan dimuat ke komponen **div** yang mempunyai *id view-list-agenda*. Berikut adalah *source code* dari *form* pencarian berbasis AJAX:

```

<fieldset>
  <form action="" method="POST">
    Nama : <input id="input-nama" type="text" name="nama" size="100"/>
    <input id="btn-form-close" type="button" value="Batal" />
  </form>
</fieldset>
<script type="text/javascript">

  // menutup form ketika tidak akan digunakan
  $("#btn-form-close").click(function(){
    $('#view-form').html("");
  });

  // melakukan proses pencarian ketika mengetikkan nama agenda
  $('#input-nama').keyup(function(){
    var nama = $('#input-nama').val();
    $.ajax({
      type:"POST",
      url:"<?php echo site_url('ajaxsample/proses_cari_agenda');?>",
      data:'nama='+nama,
      success:function(html){
        $('#view-list-agenda').html(html);
      }
    });
  });
</script>

```

Simpanlah *source code* diatas dengan nama **form_cari_agenda.php** kemudian simpan di **application -> views -> ajaxsample**. Kemudian coba tampilkan *form* tersebut dan cobalah mencari salah satu agenda yang sudah disimpan di *database*. Berikut adalah contoh penggunaan dari *form* pencarian berbasis AJAX :

Daftar Hadir Praktikum

localhost/pelatihanbasdat/index.php/ajaxsample

Daftar Agenda

[Lihat Semua](#) | [Tambah Agenda](#) | [Cari Agenda](#)

Nama :

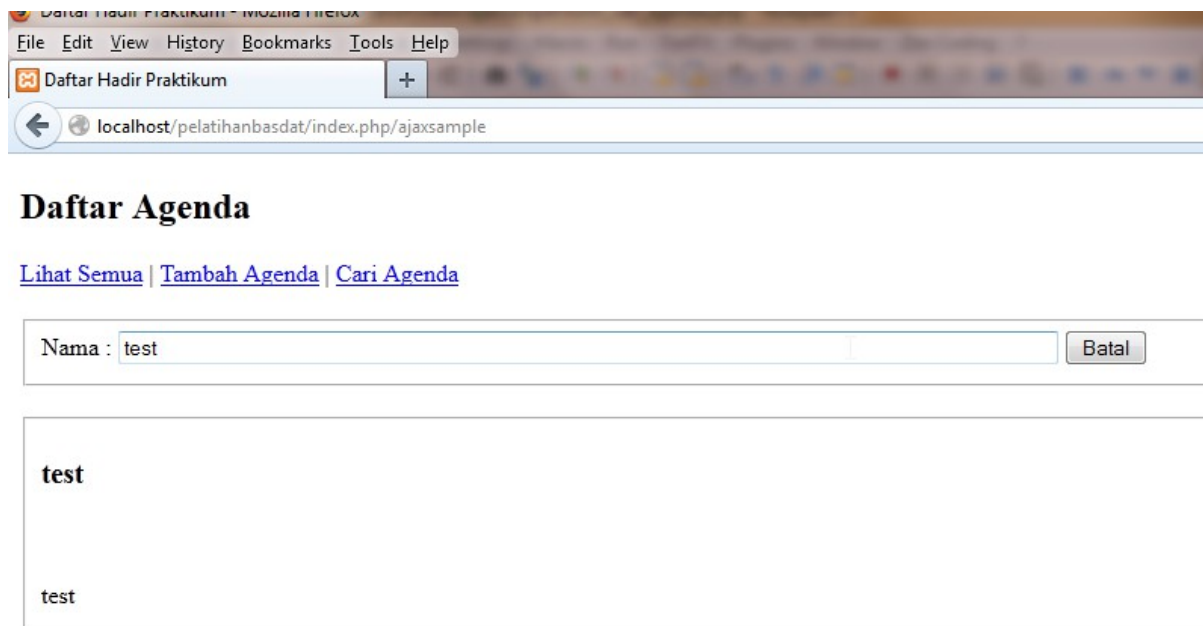
test

test

Pelatihan Basdat 2014

Daftar Hadir Praktikum

Gambar 10.4 Melihat Form Pencarian



Gambar 10.5 Melihat Hasil Pencarian

10.4. Menambahkan Fitur Tambah Agenda

Di fitur tambah agenda ini, *function* untuk memunculkan *form* tambah agenda yang terdapat di *function* **tambah_agenda()** dan memproses nilai – nilai dari *form* tambah agenda yang terdapat di *function* **proses_tambah_agenda()** tidak terlalu berbeda jauh. Hanya saja berbeda dalam memanggil *view* dan cara memproses *form* tambah agenda.

Berikut adalah *function* **tambah_agenda()** dan **proses_tambah_agenda()** di *controller* **ajaxsample**:

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

.....

// proses untuk memproses tambah agenda
public function tambah_agenda(){
    $this->load->view('ajaxsample/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);
}
```

```

}

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

```

Tentunya *form* tambah agenda akan digunakan sebagai antarmuka antara *user* dan sistem. Bila pada cara tanpa AJAX kita memerlukan atribut **action** di dalam *tag form*. Kini hal tersebut ditangani oleh AJAX JQuery sehingga tidak perlu terjadi proses memuat ulang seluruh halaman *web*.

Form tambah agenda akan memproses tambah agenda ketika terjadi *event submit()*. *Event* tersebut akan terjadi ketika *user* menekan tombol *submit*. Dengan menggunakan AJAX, data akan dikirimkan terlebih dahulu ke *function* yang dituju di *controller* kemudian jika hal tersebut berhasil maka lakukan aksi terkait saat proses tambah agenda berhasil. Dalam hal ini jika proses tambah agenda berhasil maka daftar agenda akan dimuat ulang.

Berikut adalah *source code* dari *form* tambah agenda:

```

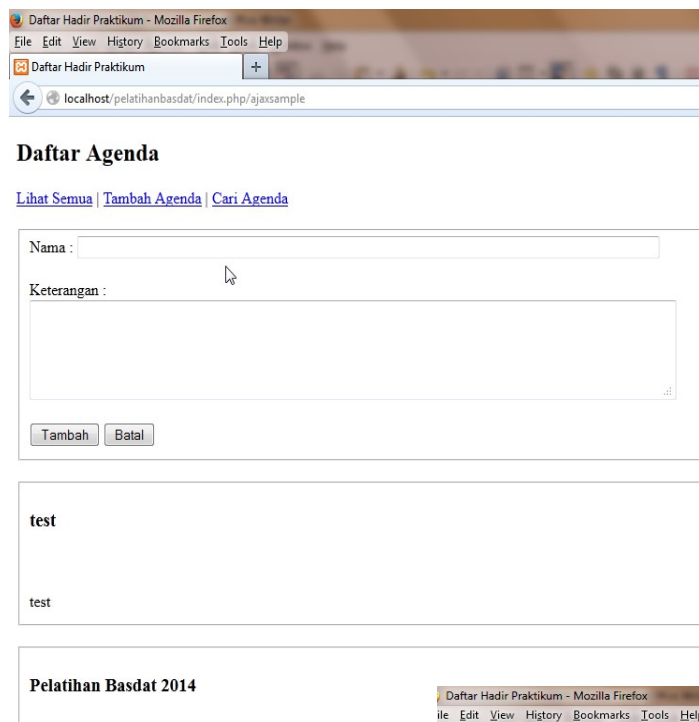
<fieldset>
  <form id="form-tambah-agenda">
    Nama : <input name="nama" size="100"/>
    <br/><br/>
    Keterangan : <br/><textarea name="keterangan" cols="83" rows="5"></textarea>
    <br/><br/>
    <input type="submit" value="Tambah" />
    <input id="btn-form-close" type="button" value="Batal" />
  </form>
</fieldset>
<script type="text/javascript">
  $(document).ready(function(){
    // menutup form ketika tidak akan digunakan
    $("#btn-form-close").click(function(){
      $('#view-form').html("");
    });

    // melakukan proses tambah agenda ketika tombol ditekan
    $('#form-tambah-agenda').submit(function(){
      $('#view-form').empty();
      $.ajax({
        type:'POST',
        url:"<?php echo site_url('ajaxsample/proses_tambah_agenda') ?>",
        data:$(this).serialize(),
        success:function (data) {
          $('#view-list-agenda').load("<?php echo site_url('ajaxsample/lihat_agenda') ?>");
        },
        error:function (data){
          $('#view-list-agenda').html('<h1>Penambahan agenda gagal !!</h1>');
        }
      });
      return false;
    });
  });

```

```
</script>
```

Simpan *source code* diatas dengan nama **form_tambah_agenda.php** dan taruh di *folder application -> views -> ajaxsample*. Kemudian coba akses *form* tersebut melalui menu **tambah agenda** dan cobalah buat agenda baru:



Daftar Hadir Praktikum - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Daftar Hadir Praktikum

localhost/pelatihanbasdat/index.php/ajaxsample

Daftar Agenda

[Lihat Semua](#) | [Tambah Agenda](#) | [Cari Agenda](#)

Nama :

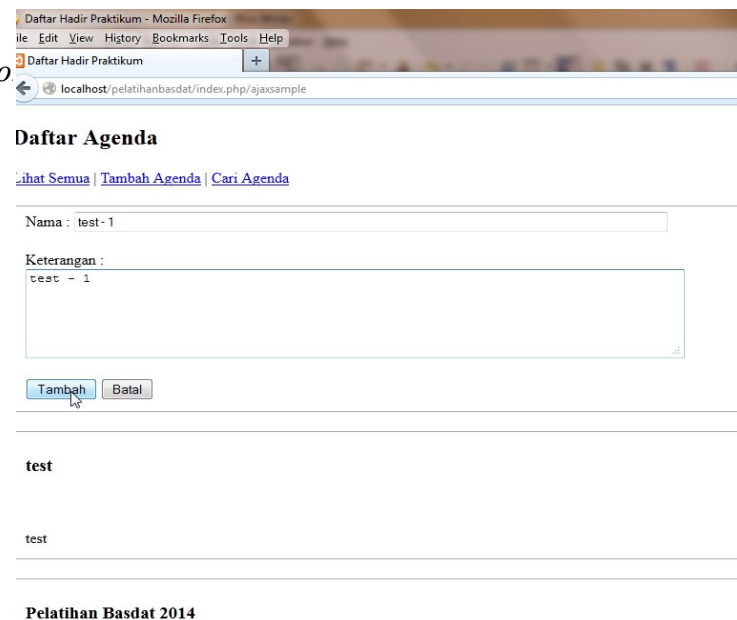
Keterangan :

test

test

Pelatihan Basdat 2014

Gambar 10.6 Menampilkan Fo



Daftar Hadir Praktikum - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Daftar Hadir Praktikum

localhost/pelatihanbasdat/index.php/ajaxsample

Daftar Agenda

[Lihat Semua](#) | [Tambah Agenda](#) | [Cari Agenda](#)

Nama : test - 1

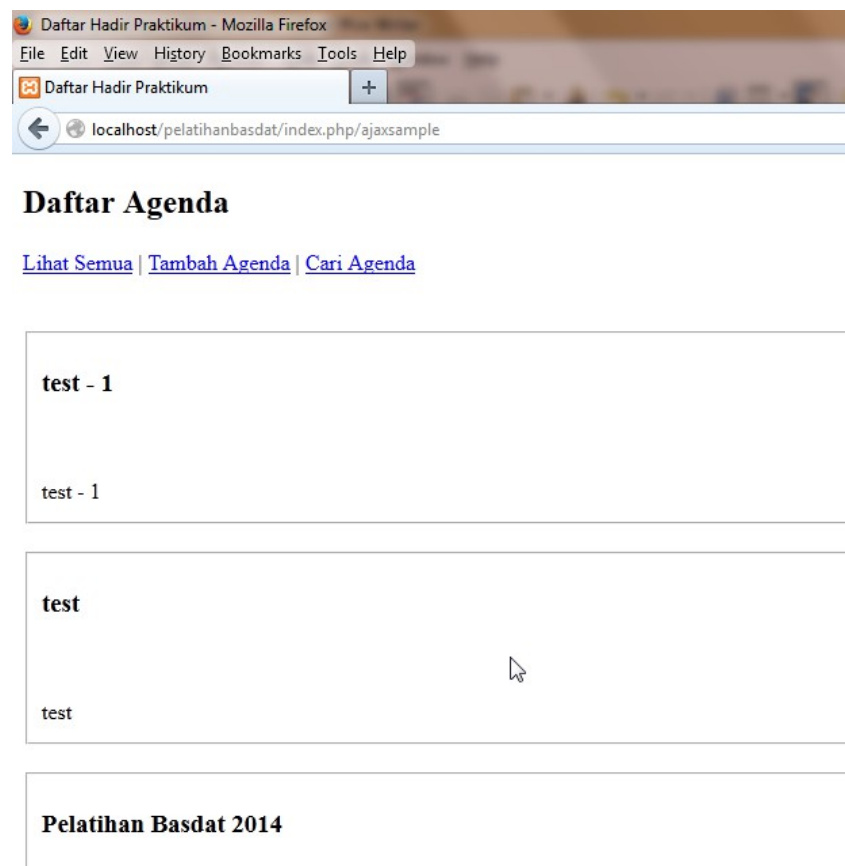
Keterangan : test - 1

test

test

Pelatihan Basdat 2014

Gambar 10.7 Membuat Agenda Baru



Gambar 10.8 Agenda Baru Berhasil Ditambahkan