PRAKTIK PEMROGRAMAN WEB

Fitri, S.T., M.Sc
Mahdiawan Nurkholifah, S.Kom
Andri Nofiar.Am, M.Kom

Praktik
PEMOGRAMAN WEB

Fitri, S.T., M.Sc.
Mahdiawan Nurkholifah, S.Kom
Andri Nofiar.Am, M.Kom

CV. KIREINARA
Publishing, Distributting & Training

Praktik
Pemrograman Web

Penulis :
Fitri, S.T., M.Sc.
Mahdiawan Nurkholifah, S.Kom
Andri Nofiar.Am, M.Kom

Editor :
Muhamad Irsadul Ngibad, S.T., M.M.

Layout :
Ammar

Cover :
Asyah Kayla

ISBN :
978-623-5745-78-7

Cetakan Pertama : Mei 2023

Penerbit :

CV.Kireinara
Jl. Dahlia No. 90
Perum. Pesona Bumi Mandiri 2
RT 06 RW 03 Beran Tambaharjo Pati
Email : redaksi.kireinara@gmail.com
Website : www.cv-kireinara.com

KATA PENGANTAR

Puji dan syukur atas kehadirat Allah SWT, atas segala limpahan taufiq serta hidayah-Nya yang telah memberi penulis kesempatan untuk menyelesaikan buku berjudul Praktik Pemrograman Web.

Dalam proses pembuatan buku ini, tentunya penulis mendapat bimbingan, arahan, koreksi dan saran. Untuk itu penulis mengucapkan terima kasih kepada Bapak Fitri,ST.,M.Sc selaku Ketua Program Studi Teknik Informatika Politeknik Kampar.

Penulis menyadari bahwa baik dari segi penulisan maupun isi, buku ini masih memiliki kekurangan, oleh karena itu penulis sangat mengharapkan kritik yang membangun dan saran dari pembaca agar terbentuknya kesempurnaan buku ini. Atas partisipasinya penulis mengucapkan terima kasih.

Kampar, Mei 2023

(Tim Penulis)

DAFTAR ISI

CHAPTER 1 : MENGENAL FRAMEWORK DAN MVC

A. Penjelasan Framework

Framework merupakan sebuah kerangka kerja yang siap digunakan yang mampu mempermudah dan mempercepat suatu pekerjaan seseorang(Rasikhah & Adriansyah, 2022). Atau bila dikaitkan dengan bahasa pemrograman framework adalah sebuah kerangka kerja yang terstruktur yang mampu mempermudah seorang programer dalam membangun atau mengambangkan sebuah perangkat lunak atau aplikasi.

Dengan menggunakan framework kita tidak perlu menulis kode script dari nol, Jadi kita cukup membuat function dan class yang sesuai dengan kebutuhan kita. Dalam framework sudah terdapat core script yang berfungsi untuk mapping class dan function yang sudah kita buat.

Alasan mengapa menggunakan framework dapat mempermudah pekerjaan:

- Mempercepat dan mempermudah membangun sebuah aplikasi berbasis web.
- Relatif memudahkan proses maintenance karena sudah ada pola tertentu dalam sebuah framework.
- Framework menyediakan fasilitas-fasilitas yang umum dipakai sehingga tidak perlu membangun dari awal.
- Lebih bebas dalam pengembangan.

Sebagai developer, tentu harus mengetahui tujuan dari penggunaannya untuk kepentingan pembuatan aplikasi. Sehingga, untuk proses pengerjaan aplikasi dapat dilakukan dengan menggunakan framework yang tepat dan sesuai dengan kebutuhan project. Berikut merupakan beberapa fungsi kerangka kerja dalam web development.

a. Kode program lebih terstruktur

Fungsi framework yang utama adalah membuat source code menjadi lebih terstruktur(Syafitri dkk., 2021). Terstruktur disini, berarti program yang dibuat akan dimasukkan ke dalam setiap komponen sesuai dengan fungsi nya masing –

masing. Terdapat tiga komponen utama untuk mengembangkan website menggunakan model framework tersebut yakni menggunakan konsep paradigma MVC (*Model, View, Controller*)(Bagus Wibisono, 2020).

Model berfungsi untuk tempat atau wadah menampung kode program berupa algoritma pemrograman dan penghubung database aplikasi. View berfungsi sebagai wadah menampung kode program untuk membuat tampilan yang nantinya ditampilkan kepada customer / client. Dan controller berfungsi untuk menghubungkan model dan view agar menjadi sebuah website secara keseluruhan(Ridwan dkk., 2022).

b. Membantu kinerja dari developer

Fungsi yang kedua adalah membantu kinerja dari developer sendiri. Dari sini, anda pasti sudah berpikir bahwa sebenarnya dalam membuat sebuah aplikasi dapat dilakukan tanpa menggunakan bantuan framework. Tentu saja, bisa dilakukan jika proyek yang ditangani dalam lingkup kecil.

Apabila anda membuat aplikasi atau tampilan website untuk sebuah perusahaan atau organisasi besar tentu saja hal tersebut tidak disarankan bahkan mindset tersebut harus segera diubah. Framework diciptakan untuk memudahkan kinerja dari developer dalam segi efisiensi waktu serta resource yang dibutuhkan.

c. Meningkatkan keamanan perangkat lunak / website

Fungsi yang kedua adalah membantu kinerja dari developer sendiri. Dari sini, anda pasti sudah berpikir bahwa sebenarnya dalam membuat sebuah aplikasi dapat dilakukan tanpa menggunakan bantuan framework. Tentu saja, bisa dilakukan jika proyek yang ditangani dalam lingkup kecil.

Apabila anda membuat aplikasi atau tampilan website untuk sebuah perusahaan atau organisasi besar tentu saja hal tersebut tidak disarankan bahkan mindset tersebut harus segera diubah. Framework diciptakan untuk memudahkan kinerja dari developer dalam segi efisiensi waktu serta resource yang dibutuhkan.

d. Pemeliharaan dan dokumentasi akan lebih mudah

Dengan melakukan maintenance, anda dapat mengubah versi website tersebut dan menambahkan beberapa fitur dengan lebih mudah dan aman.

Sehingga, ketika ada perbaikan pada website maka terdapat notifikasi atau pesan bagi pengguna bahwa website masih dalam perbaikan.

Selanjutnya, dari segi dokumentasi juga lebih terstruktur. Anda akan sangat kerepotan apabila dalam proses dokumentasi tidak menggunakan bantuan framework. Dalam kerangka kerja, setiap dokumen aplikasi yang dibangun dapat diidentifikasi dengan mudah dan cepat.

e. Mempercepat proses pembuatan aplikasi

Fungsi terakhir adalah untuk mempercepat dalam proses pembuatan website. Dalam hal ini, bukan berarti pembuatan website yang baik dikerjakan dengan cepat saja. Tetapi, proses pembuatan dapat dilakukan dengan lebih cepat dan menghasilkan produk yang berkualitas dan sesuai dengan kebutuhan customer.

Developer dapat mengembangkan aplikasi dengan menggunakan komponen–komponen yang telah tersedia dalam framework. Sehingga, tidak perlu untuk menyusun ulang dari awal kode program.

B. Penjelasna MVC

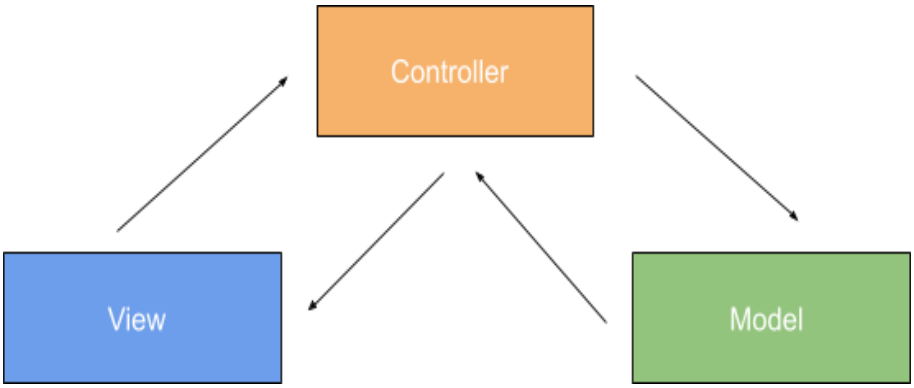
MVC adalah singkatan dari “Model View Controller”merupakan suatu konsep yang sangat populer dalam pembangunan website dan aplikasi(Chusyairi dkk., 2020). MVC memisahkan pengembangan aplikasi berdasarkan 3 jenis komponen utama yaitu manipulasi data, user interface, dan bagian yang menjadi kontrol aplikasi. Komponen-komponen utama tersebut membangun suatu MVC pattern atau bagian yang diberi nama **Model**, **View** dan **Controller**(Rahman, 2023)**.**

- **Model,** bagian yang mengelola dan berhubungan langsung dengan database;
- **View**, bagian yang akan menyajikan tampilan informasi kepada pengguna;
- **Controller**, bagian yang menghubungkan model dan view dalam setiap proses request dari user.

Dengan menggunakan konsep MVC, suatu aplikasi dapat dikembangkan sesuai dengan kemampuan PIC-nya, Developer yang menangani bagian Model dan Controller, sedangkan Web Designer yang menangani bagian View, sehingga penggunaan arsitektur MVC dapat meningkatkan maintainability dan

pengorganisasian kode(Alelo dkk., 2021). Meskipun begitu, namun tetap dibutuhkan komunikasi yang baik Developer dan Web Designer dalam menangani variabel dan parameter data yang ada.

Oke, setelah mempelajari apa itu MVC, sekarang saatnya memahami bagaimana alur kerja dari MVC menurut (Hapsari, 2020). Mari lihat bagan berikut ini:



1. Bagian **view** akan merequest informasi untuk bisa ditampilkan kepada pengguna.
2. Request tersebut kemudian diambil oleh **controller** dan diserahkan bagian **model** untuk diproses;
3. Model akan mengolah dan mencari data informasi tersebut di dalam database;
4. Model memberikan kembali pada controller untuk ditampilkan hasilnya di view;
5. Controller mengambil hasil olahan yang dilakukan di bagian model dan menatanya di bagian view.

Alur kerja MVC dalam sistem website sebenarnya cukup sederhana seperti ditunjukkan pada bagan di atas. Namun, kalau penjelasannya masih terlalu teknis, begini analoginya:

Sekarang, anggaplah Anda sedang berada di sebuah restoran. Dalam konsep MVC ini, Anda adalah **view,** pelayan adalah **controller**, dan chef adalah **model.**

Ketika Anda memesan salah satu menu, pelayan akan mencatat pesanan Anda dan memberikannya pada chef. Setelah itu, chef akan mencari bahan yang diperlukan di kulkas (database) dan mulai memasaknya untuk Anda.

Setelah selesai dimasak, chef akan memberikan pada pelayan untuk diantarkan pada Anda.

Nah, seperti itulah cara kerja MVC pada setiap bagiannya. Lebih mudah dipahami?

CHAPTER 2 : FRAMEWORK CODEIGNITER 3

A. Pengertian Codeigniter

CodeIgniter adalah sebuah web application network yang bersifat open source yang digunakan untuk membangun aplikasi php dinamis (Irawan dkk., 2020). CodeIgniter menjadi sebuah framework PHP dengan model MVC (Model, View, Controller) untuk membangun website dinamis dengan menggunakan PHP yang dapat mempercepat pengembang untuk membuat sebuah aplikasi web (Rahman, 2023). Selain ringan dan cepat, CodeIgniter juga memiliki dokumentasi yang super lengkap disertai dengan contoh implementasi kodenya. Dokumentasi yang lengkap inilah yang menjadi salah satu alasan kuat mengapa banyak orang memilih CodeIgniter sebagai framework pilihannya. Karena kelebihan-kelebihan yang dimiliki oleh CodeIgniter, pembuat PHP Rasmus Lerdorf memuji CodeIgniter di frOSCon (Agustus 2008) dengan mengatakan bahwa dia menyukai CodeIgniter karena “it is faster, lighter and the least like a framework.”

CodeIgniter pertama kali dikembangkan pada tahun 2006 oleh Rick Ellis (Fadilah dkk., 2020). Dengan logo api yang menyala, CodeIgniter dengan cepat “membakar” semangat para web developer untuk mengembangkan web dinamis dengan cepat dan mudah menggunakan framework PHP yang satu ini.

B. Fungsi Codeigniter

1. Mempercepat dan mempermudah kita dalam pembuatan website.
2. Menghasilkan struktur pemrograman yang sangat rapi, baik dari segi kode maupun struktur file phpnya.
3. memberikan standar coding sehingga memudahkan kita atau orang lain untuk mempelajari kembali system aplikasi yang dibangun.

C. Fitur Codeigniter

CodeIgniter juga memiliki fitur-fitur berguna yang membantu developer membuat sebuah website. Berikut adalah beberapa fitur utama yang ada pada framework menurut (Fadllullah dkk., 2022):

1. Kompatibel dengan banyak jenis database

2. Menyediakan query builder support.
3. CodeIgniter bersifat Independent.
4. Mengamankan website Anda dari cross site scripting.
5. Menyediakan validasi form/data dan juga session management.

D. Keunggulan Codeigniter

Selain mempermudah kinerja web developer, framework CodeIgniter juga memiliki banyak keunggulan lain loh Golden friends! Berikut adalah beberapa manfaat dan keunggulannya menurut (Syafitri dkk., 2021):

- **Ringan** — seluruh framework CodeIgniter mempunyai library dan resources yang sangat ringan. Bahkan Anda bisa mendownload versi terbaru framework ini dengan ukuran file kurang dari 1MB.
- **Performa cepat** — saat ini waktu loading rata-rata dari framework ini adalah kurang dari 50ms. Tentunya ini performa yang sangat cepat dan disukai banyak developer.
- **Minim konfigurasi** — framework ini pun terbilang mempunyai konfigurasi yang sangat mudah dan sederhana, developer hanya perlu melakukan sedikit pengaturan.
- **Banyak support dan komunitas** — bersifat open source, frame CodeIgniter memiliki banyak komunitas pendukung yang terdiri dari banyak web developer dari seluruh dunia.
- **Dokumentasi yang lengkap dan informatif** — framework ini pun memiliki dokumentasi official yang sangat lengkap. Anda bisa mempelajari semua hal yang perlu Anda ketahui dengan user manual yang sudah disediakan oleh CodeIgniter.
- **Maintenance yang mudah** — komponen CodeIgniter dapat bekerja secara mandiri tanpa bergantung dengan komponen lainnya. Tentunya hal ini sangat memudahkan maintenance atau pemeliharaan website nantinya.
- **Fitur khusus** — CodeIgniter menyediakan beberapa fitur khusus yang tidak banyak dimiliki framework lainnya seperti fitur mengirim email, manajemen database, dan juga manajemen session.

CHAPTER 3 : KONFIGURASI DAN INSTALASI

A. Persiapan belajar CodeIgniter 3

Berikut ini beberapa peralatan yang harus kamu siapkan di komputermu:

1. Teks Editor
2. Web Browser
3. Web Server: PHP, MySQL, Phpmyadmin
4. File Project Codeigniter

Mari kita siapkan satu-per-satu.

1. Teks Editor

Teks editor akan kita gunakan untuk menulis kode. Kamu bebas menggunakan teks editor apa saja untuk *coding* CI.

2. Web Browser

Web browser akan kita gunakan untuk melihat hasil dari aplikasi. Kamu juga bebas menggunakan web browser apapun, asalkan masih mendukung teknologi web modern zaman sekarang.

Rekomendasi, gunakan Google Chrome atau Firefox.

3. Web Server

Codeigniter merupakan framework PHP, karena itu ia pasti membutuhkan web server. Berikut ini requirement server untuk Codeigniter 3:

- PHP Versi 7.0+
- MySQL Versi 5.1+
- Phpmyadmin

Jika kamu sudah menginstal XAMPP, maka ketiga aplikasi server ini sudah terpenuhi. Tapi jika kamu pengguna Linux, maka ini bisa diinstal satu-per-satu.

4. File Project Codeigniter

File project Codeigniter dapat di-download di website resmi Codeigniter. Nanti kita akan mendapatkan file berupa ZIP. File inilah yang akan kita gunakan untuk mulai membuat proyek Codeigniter.

B. Cara Membuat Project CodeIgniter 3

Cara Membuat Virtual Host di XAMPP Windows

Untuk mengakses suatu project website di *local environment* Windows, Anda mungkin membutuhkanXAMPP yang digunakan sebagai web server.

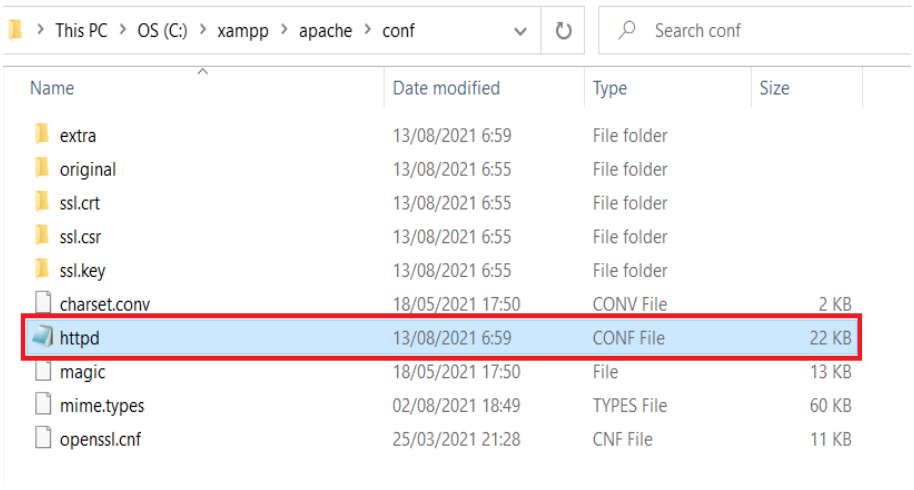
Ketika XAMPP dijalankan, Anda perlu mengakses URL localhost/namaprojectanda di web browser untuk menampilkan project seperti terlihat pada gambar di bawah ini:

Dengan virtual host Windows, Anda bisa mengubah nama localhost menjadi nama unik sesuai dengan keinginan Anda. Contohnya, nama **hostbaru.net** yang akan kami gunakan pada tutorial ini.

4 langkah yang bisa Anda ikuti untuk membuat virtual host XAMPP adalah sebagai berikut:

1. Mengedit File httpd.conf di XAMPP

Langkah pertama dalam tutorial cara membuat virtual host di XAMPP adalah mengedit file httpd.conf yang dapat Anda temukan pada folder C:\xampp\apache\conf.



Bukalah file **httpd.conf** menggunakan text editor yang Anda miliki. Pada tutorial virtual host XAMPP kali ini, kami menggunakan aplikasi Notepad.

Jika file sudah terbuka, temukan section pengaturan **virtual hosts** seperti pada gambar berikut ini:

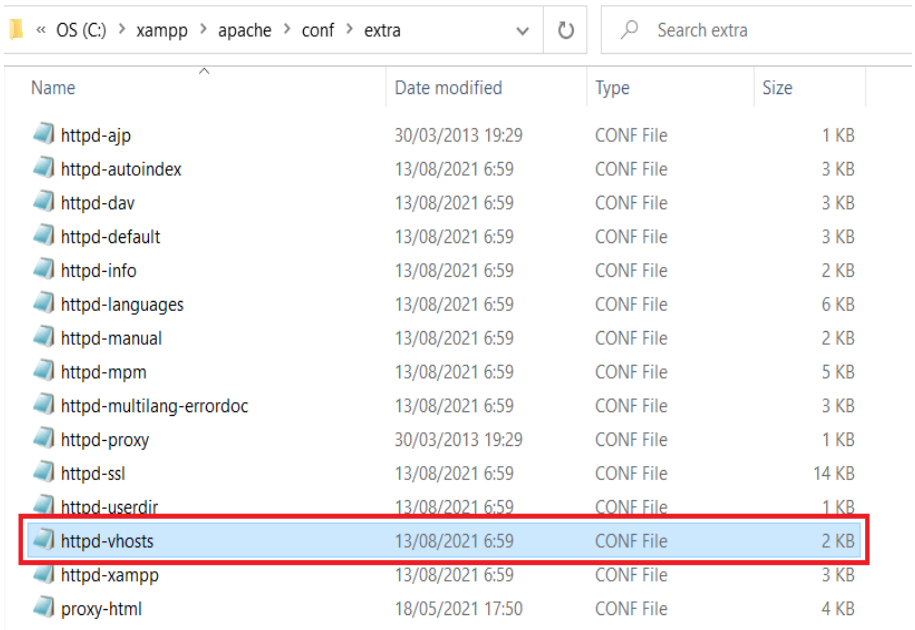
Selanjutnya, hapus tanda pagar (#) pada baris **Include conf/extra/httpd-vhosts.conf** menjadi seperti ini:

```
# Virtual hosts
Include conf/extra/httpd-vhosts.conf
```

Kalau sudah dihapus, Anda bisa menyimpan perubahan dengan cara tekan tombol **CTRL + S** di keyboard. Kemudian, tutup file tersebut.

2. Mengedit File https-vhosts.conf di XAMPP

Berikutnya, Anda perlu membuat virtual host pada file httpd-vhosts.conf. Caranya, masuklah ke folder C:\xampp\apache\conf\extra dan bukalah file httpd-vhosts.conf berikut ini:



Kemudian, tambahkan script di bawah ini pada bagian akhir file tersebut:

```
<VirtualHost *:80>
    ServerAdmin webmaster@belajarcodeigniter.test
    DocumentRoot "D:\vhost\belajarcodeigniter.test"
    ServerName belajarcodeigniter.test
    ErrorLog "D:\vhost\belajarcodeigniter.test-error.log"
    CustomLog "D:\vhost\belajarcodeigniter.test-access.log" common
    <Directory "D:\vhost\belajarcodeigniter.test">
        Options Indexes FollowSymLinks Includes ExecCGI
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

yang harus anda ubah dan anda sesuaikan dengan domain anda:

ServerAdmin : webmaster@namadomain_anda.

DocumentRoot : lokasi anda menyimpan file project

Servername : namadomain_anda

ErrorLog : lokasi untuk menyimpan file log error website anda

CustomLog : lokasi untuk menyimpan file log access website anda

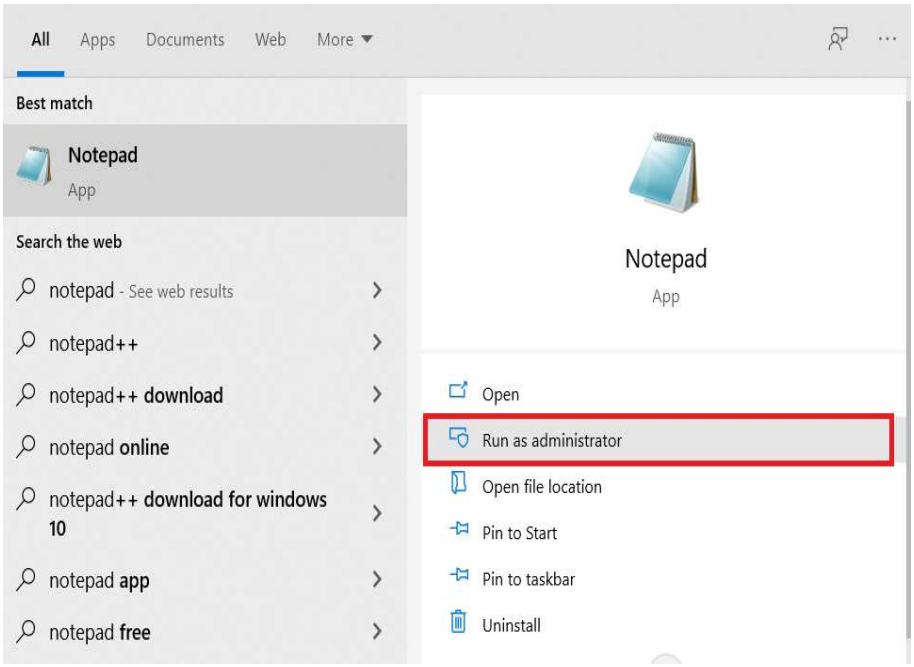
Directory : lokasi anda menyimpan file project

Jangan lupa untuk menyimpan perubahan yang sudah dilakukan, lalu klik tombol **Close**.

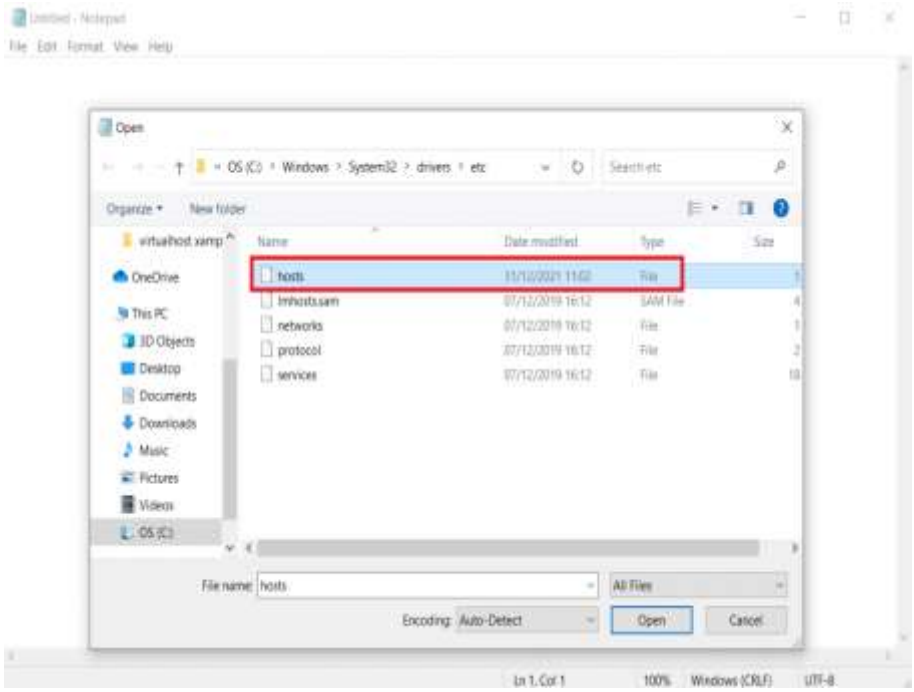
3. Mengedit File hosts di Windows

Langkah selanjutnya dalam cara membuat virtual host di XAMPP adalah mengedit file hosts. File ini merupakan file Windows yang berada pada direktori C:\Windows\System32\drivers\etc.

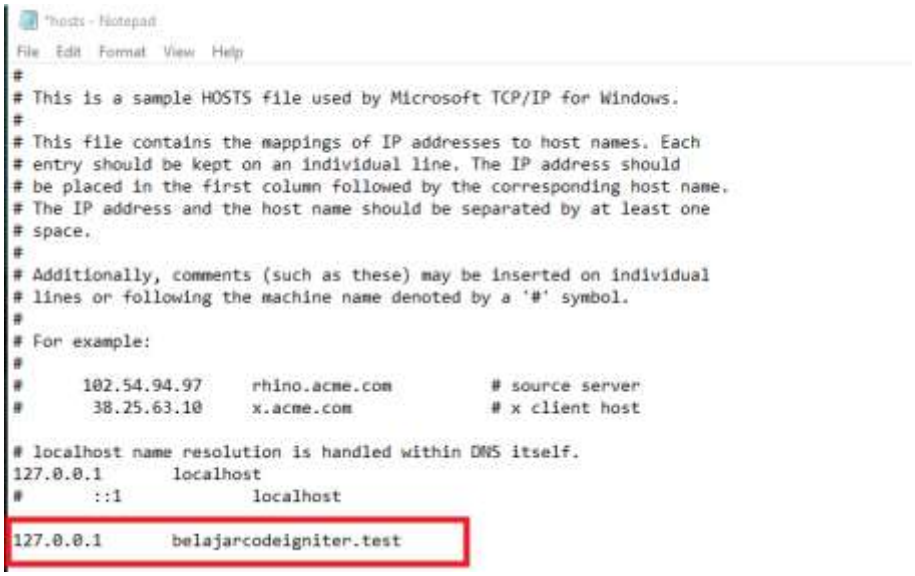
Untuk mengedit file hosts windows, Anda membutuhkan akses sebagai administrator. Oleh karena itu, silakan buka notepad dengan menggunakan akses Run as administrator terlebih dulu sebelum membuka file hosts.



Kemudian, pilih menu File > Open dan masuklah ke direktori C:\Windows\System32\drivers\etc. Klik file hosts, lalu klik tombol Open untuk membuka file hosts.



Jika file **hosts** sudah terbuka, tambahkanlah nama host baru Anda pada bagian akhir file seperti pada gambar di bawah ini:



Tekan tombol **CTRL + S** untuk menyimpan perubahan, kemudian tutup file hosts tersebut.

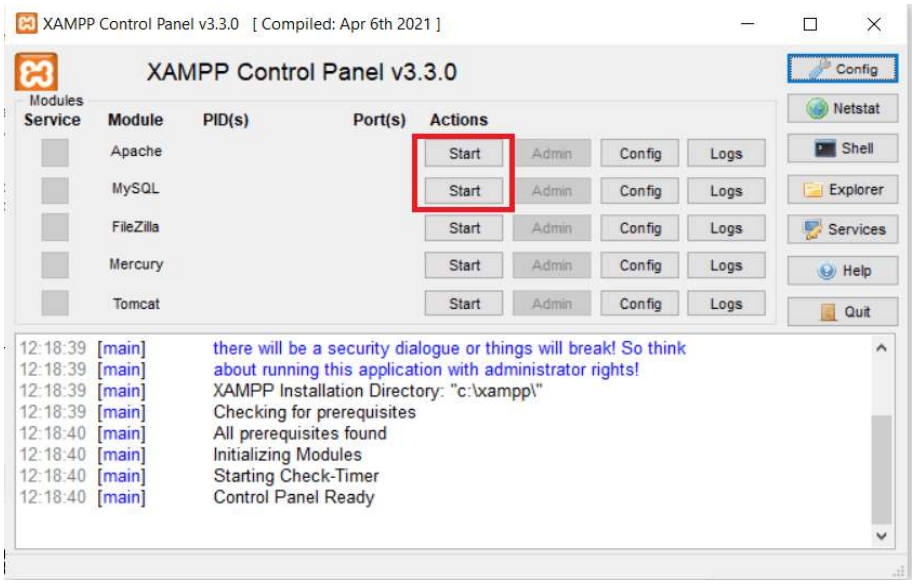
4. Membuat Folder Project

Buatlah satu folder untuk menampung project anda seperti struktur folder berikut:

```
Local Disk (D:)                  ==> tempat disk penyimpanan
------ vhost                     ==> folder vhost
--------belajarcodeigniter.test  ==> folder untuk menampung file project (nama folder harus sesuai dengan domain)
```

5. Merestart XAMPP

Terakhir, lakukanlah restart pada XAMPP. Caranya, klik tombol Start pada modul Apache dan MySQL seperti berikut ini:



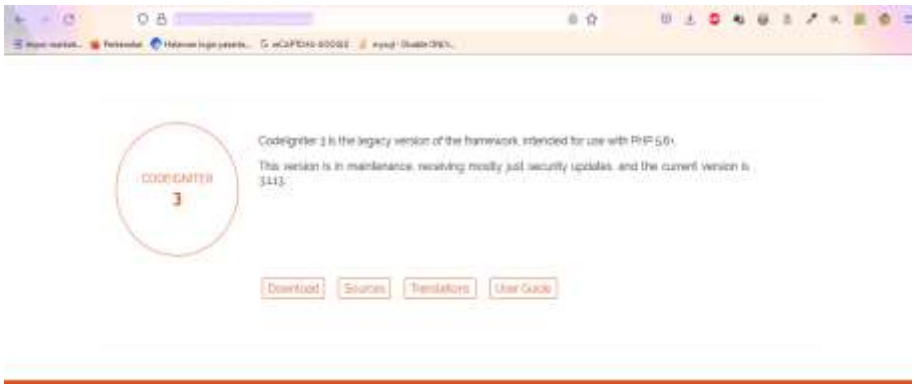
Kemudian, bukalah web browser dan akses nama host baru Anda. Harusnya, sekarang Anda sudah berhasil mengaksesnya tanpa error seperti pada gambar berikut ini:

Kenapa hasilnya seperti itu? karena pada folder project (belajarcodeigniter.test) tidak ada file index.php atau index.html.

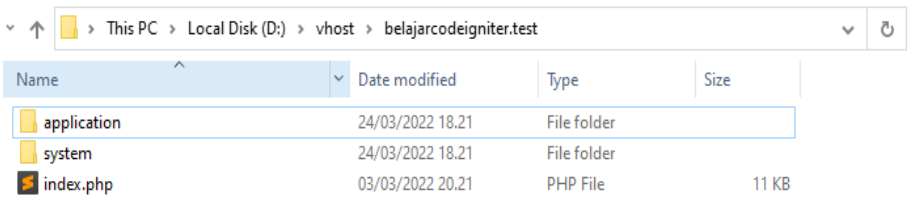
Instalasi Project Codeigniter

Langkah-langkah yang harus dilakukan untuk membuat project CI:

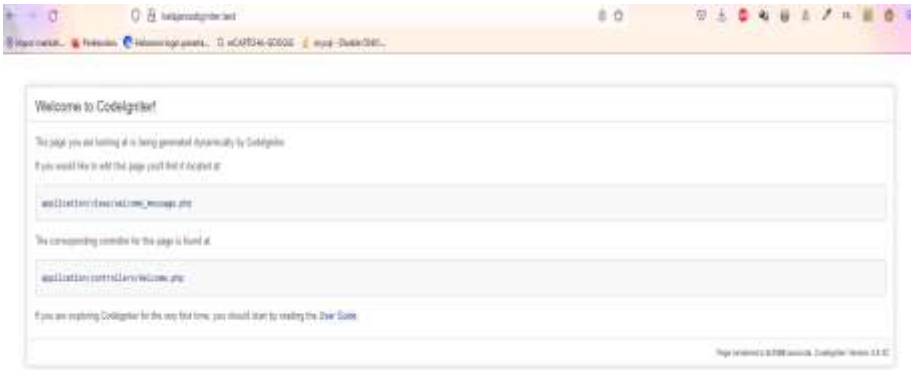
1. Download Codeigniter;
2. Ekstrak File ZIP Codeigniter ke htdocs.



Silahkan buka **https://codeigniter.com/download** untuk mendownload. Kita akan mendapatkan sebuah file zip **bcit-ci-CodeIgniter-3.1.13-0-gbcb17eb.zip**, ekstrak file tersebut ke dalam **D:\vhost\belajarcodeigniter.test.** berikut struktur file dan folder yang akan ada di folder belajarcodeigniter.test :



Jika sudah, coba buka web browser dan buka alamat **belajarcodeigniter.test**



CHAPTER 4 : CODEIGNITER URLs DAN URI ROUTING

A. Memahami CodeIgniter URLs

Secara default, URL di CodeIgniter dirancang untuk mesin pencari dan mudah digunakan. Daripada menggunakan pendekatan "string kueri" standar untuk URL yang identik dengan sistem dinamis, CodeIgniter menggunakan pendekatan berbasis segmen seperti berikut:

```
example.com/news/article/my_practice
```

Note :
URL string kueri dapat diaktifkan secara opsional, seperti yang dijelaskan di bawah ini.

URI Segments

Segmen dalam URL biasanya mengikuti pendekatan Model-View-Controller, Perhatikan URL segmen berikut:

```
example.com/class/function/ID
```

Keterangan URL Segmen di atas :

- Segmen pertama mewakili kelas di controller yang di panggil.
- Segmen kedua mewakili fungsi atau method dalam class controller.
- Segemen ketiga merupakan segmen tambahan, yang mewakili ID dan variabl apa saja yang akan diteruskan ke class controller.

Library URI dan Helper URL berisi fungsi-fungsi yang memudahkan untuk bekerja dengan data URI Anda. Selain itu, URL Anda dapat dipetakan ulang menggunakan fitur Perutean URI agar lebih fleksibel.

Penambahan URL Suffix

Dalam file config/config.php, Anda dapat menentukan sufiks yang akan ditambahkan ke semua URL yang dihasilkan oleh CodeIgniter. Misalnya, jika URL adalah ini:

```
example.com/index.php/products/view/shoes
```

Anda juga dapat menambahkan sufiks, seperti **.html**, agar halaman tampak seperti jenis tertentu:

```
example.com/index.php/products/view/shoes.html
```

Enabling Query Strings

Dalam beberapa kasus, Anda mungkin lebih suka menggunakan URL string kueri seperti berikut:

```
index.php?c=products&m=view&id=345
```

CodeIgniter secara opsional mendukung kemampuan ini, yang dapat diaktifkan di file application/config.php. Jika Anda membuka file konfigurasi Anda, Anda akan melihat item ini:

```
$config['enable_query_strings'] = FALSE;
$config['controller_trigger'] = 'c';
$config['function_trigger'] = 'm';
```

Jika Anda mengubah "**enable_query_strings**" menjadi **TRUE**, fitur ini akan aktif. Pengontrol dan fungsi Anda kemudian akan dapat diakses menggunakan kata-kata "**trigger**" yang telah Anda atur untuk memanggil pengontrol dan metode Anda:

```
index.php?c=controller&m=method
```

Note :

Jika Anda menggunakan string kueri, Anda harus membuat URL Anda sendiri, daripada menggunakan URL helper (dan helper

lain yang menghasilkan URL, seperti beberapa form helper) karena ini dirancang untuk bekerja dengan URL berbasis segmen.

B. Memahami URL Routing

Router pada Codeigniter bertugas untuk menentukan ***controller*** dan ***method***/fungsi yang akan dieksekusi.

Ketika kita membuka, **http://belajarcodigniter.test/** maka fungsi yang akan dieksekusi adalah fungsi **index()** yang berada di dalam ***controller*** **welcome**.

```
class Welcome extends CI_Controller {
    public function index()
    {
        $this->load->view('welcome_message');
    }
}
```

Kenapa bisa begitu?

Ini karena konfigurasi router defaultnya adalah *controller* welcome.

Perhatikan code pada gambar berikut:

```
$route['default_controller'] = 'frontend';
$route['404_override'] = '';
$route['translate_uri_dashes'] = FALSE;
```

Fungsi **index()** adalah fungsi yang akan dieksekusi saat kita mengakses ***controller*** **welcome**.

Biasanya ada hubungan satu-ke-satu antara string URL dan kelas/metode pengontrol yang sesuai. Segmen dalam URI biasanya mengikuti pola ini:

example.com/class/function/id/

Namun, dalam beberapa kasus, Anda mungkin ingin memetakan kembali hubungan ini sehingga kelas/metode yang berbeda dapat dipanggil alih-alih yang sesuai dengan URL.

Misalnya, Anda ingin URL Anda memiliki prototipe ini:

```
example.com/product/1/
example.com/product/2/
example.com/product/3/
example.com/product/4/
```

Biasanya segmen kedua dari URL dicadangkan untuk nama metode, tetapi dalam contoh di atas, ia memiliki ID produk. Untuk mengatasinya, CodeIgniter memungkinkan Anda untuk memetakan ulang pengendali URI.

Cara Menetapkan Aturan Routing Sendiri

Aturan perutean ditentukan dalam file **application/config/routes.php**. Di dalamnya akan melihat larik bernama **$route** yang memungkinkan untuk menentukan kriteria perutean sendiri. Rute dapat ditentukan menggunakan wildcard atau Ekspresi Reguler.

1. Wildcards

Tipikal Route wildcard mungkin akan terlihat seperti ini:

```
$route['product/:num'] = 'catalog/product_lookup';
```

Dalam sebuah rute, kunci larik berisi URI yang akan dicocokkan, sedangkan nilai larik berisi tujuan yang harus dirutekan ulang. Dalam contoh di atas, jika kata literal "produk" ditemukan di segmen pertama URL, dan nomor ditemukan di segmen kedua, kelas "katalog" dan metode "pencarian_produk" akan digunakan.

Anda dapat mencocokkan nilai literal atau Anda dapat menggunakan dua jenis wildcard:

Terdapat dua type route wildcards yang dapat kita gunakan unutk pencocokan nilai yakni:

- **(:num) -** akan cocok dengan segmen yang hanya berisi angka
- **(:any) -** akan cocok dengan segmen yang berisi karakter apa pun (kecuali untuk '/', yang merupakan pembatas segmen).

Contoh :

```
$route['journals'] = 'blogs';
```

URL yang berisi kata "jurnal" di segmen pertama akan dipetakan ulang ke kelas "blog".

```
$route['blog/joe'] = 'blogs/users/34';
```

URL yang berisi segmen blog/joe akan dipetakan ulang ke kelas "blog" dan metode "pengguna". ID akan disetel ke "34".

```
$route['product/(:any)'] = 'catalog/product_lookup';
```

URL yang berisi segmen blog/joe akan dipetakan ulang ke kelas "blog" dan URL dengan "produk" sebagai segmen pertama, dan apa pun di segmen kedua akan dipetakan ulang ke kelas "katalog" dan metode "pencarian_produk".

```
$route['product/(:num)'] = 'catalog/product_lookup_by_id/$1';
```

URL dengan "produk" sebagai segmen pertama, dan nomor di segmen kedua akan dipetakan ulang ke kelas "katalog" dan metode "product_lookup_by_id" meneruskan kecocokan sebagai variabel ke metode.

2. Regular Expressions

Rute RegEx yang khas mungkin terlihat seperti ini:

```
$route['products/([a-z]+)/(\d+)'] = '$1/id_$2';
```

pada contoh di atas, URI yang mirip dengan products/shirts/123 akan memanggil kelas pengontrol "shirts" dan metode "id_123".

Dengan ekspresi reguler, Anda juga dapat menangkap beberapa segmen sekaligus. Misalnya, jika pengguna mengakses area yang dilindungi kata sandi dari aplikasi web Anda dan Anda ingin dapat mengarahkan mereka kembali ke halaman yang sama setelah mereka masuk, Anda mungkin menemukan contoh ini berguna:

```
$route['login/(.+)'] = 'auth/login/$1';
```

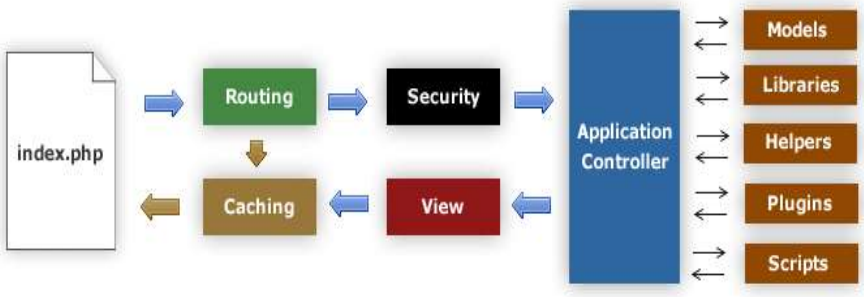
Dalam contoh di atas, jika $1 placeholder berisi garis miring, itu akan tetap dibagi menjadi beberapa parameter saat diteruskan ke Auth::login().

CHAPTER 5 : CONTROLLERS

Controller adalah sebuah file class yang diberi nama sesuai dengan nama URI atau dikaitkan dengan URI.

A. Tugas Controller

Controller adalah jantungnya aplikasi yang bertugas meng-handle HTTP request.



Perhatikan URL Berikut:

example.com/index.php/blog/

Jika kita membuka URL tersebut, maka Codeigniter akan mencari Controller bernama Blog. Lalu si Controller Blog akan meng-handle request kita. Meng-*handle* dapat kita artikan dengan:

- Menerima HTTP request;
- Memprosesnya;
- Mengirim HTTP response.

Jadi itulah tugas utama dari Controller.

Tapi ada juga yang memberi tugas Controller untuk menentukan logika bisnis.

B. Aturan Membuat Controller

1. Penulisan Nama File

   File Controller harus dibuat di dalam folder application/controllers dan penulisan nama filenya harus diawali dengan huruf besar.

   Contoh:

Penulisan yang benar

```
Blog.php
Product.php
```

Penulisan yang salah

```
blog.php
product.php
```

Jika nama terdiri dari dua suku kata atau lebih, boleh dipisah dengan *underscore*.

```
Detail_blog.php
```

2. Penulisan Nama Class

File Controller berisi sebuah class, cara menulis nama class harus diawali dengan huruf kapital. Kalau bisa ikuti nama filenya.

Contoh yang benar

```
<?php
class Blog extends CI_Controllers{

}
```

Contoh yang salah

```
<?php
class blog extends CI_Controllers{

}
```

Jika nama terdiri dari dua suku kata atau lebih, bisa dipisah dengan *underscore*.

```
<?php
class Blog_details extends CI_Controllers{


}
```

3. Penulisan Nama Method

   Nama method ditulis dengan huruf kecil dan jika terdapat lebih dari satu suku kata, bisa dipisah dengan *underscore*.

   Contoh Penulisan

```
<?php
class Blog extends CI_Controller {
 public function index()
 {
        echo 'Hello World!';
 }

 public function comments()
 {
        echo 'Look at this!';
 }
}
```

C. Passing Segmen URI ke Method

Jika URI Anda berisi lebih dari dua segmen, mereka akan diteruskan ke metode Anda sebagai parameter.

Misalnya, Anda memiliki URI seperti ini:

```
example.com/index.php/products/shoes/sandals/123
```

Metode Anda akan melewati URI segmen 3 dan 4 (“sandal” dan “123”):

contoh:

```
<?php
class Products extends CI_Controller {

        public function shoes($sandals, $id)
        {
        echo $sandals;
        echo $id;
        }
}
```

D. Input dan Output di Controllers

Pada controller, kita bisa mengambil input dari properti $this->input dan untuk menghasilkan output kita bisa gunakan echo, load->view, dan properti $this->output.

a. Empat Macam Input dasar

Ada empat macam inputan yang bisa kita dapatkan di Controller dengan properti $this->input:

1. $this->input->post() untuk mengambil input dari form yang menggunakan metode POST untuk pengiriman data;
2. $this->input->get() untuk mengambil input dari query string atau form yang menggunakan metode GET untuk pengiriman data;
3. $this->input->cookie() untuk mengambil input dari cookie browser;
4. $this->input->server() untuk mengambil input dari server;

Sebenarnya kita juga bisa memanfaatkan variabel global bawaan PHP seperti $_POST, $_GET, $_COOKIE, dan $_SERVER.

Apa bedanya variabel tersebut dengan $this->input?

Bedanya:

Mengambil input dari $this->input lebih aman, karena sudah di-filter dan diproteksi XSS serta CSRF.

b. Membuat Output

Output akan menjadi HTTP Response dari request yang diterima oleh Controller. Kita bisa membuat output dengan fungsi echo(), print(), dan fungsi output lainnya. Tapi biasanya, kita menampilkan sebuah view dengan fungsi:

```
$this->load->view('nama_view');
```

E. Private Method di Controller

Jika kita membuat method pada controller dengan modifier private, maka method tersebut tidak akan bisa diakses dair publik atau URL.

```
<?php
class Blog extends CI_Controller {
  public function index()
  {
          echo 'Hello World!';
  }

 private function _utility()
  {
          echo 'Look at this!';
  }
}
```

Coba akses controller tersebut melalui URL:

```
example.com/index.php/blog/_utility
```

Maka pasti tidak akan bisa.

Method private biasanya dipakai untuk fungsi tambahan seperti utility dan hanya bisa dipanggil pada class itu sendiri.

Mengawali nama metode dengan garis bawah juga akan mencegahnya dipanggil. Ini adalah fitur warisan yang tersisa untuk backwards-compatibility.

F. Class Constructors

Method __costruct() adalah method yang akan selalu dieksekusi setiap kita mengakses Controller, entah kita mau akses method apapun.

Biasanya method __construct() digunakan untuk inisialiasi atau persiapan awal.

Jika Anda bermaksud menggunakan konstruktor di salah satu controllers, Anda HARUS menempatkan baris kode berikut di dalamnya :

```
parent::__construct();
```

Alasan baris ini diperlukan adalah karena konstruktor lokal Anda akan menimpa konstruktor yang ada di kelas pengontrol induk sehingga kita perlu memanggilnya secara manual.

Contoh :

```
<?php
class Blog extends CI_Controller {


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

Pada contoh tersebut, kita melakukan load model pada method __construct(), ini akan membuat model post di-load otomatis pada Controller Blog.

Latihan

1. Buatlah controller dengan nama Belajar.
2. Buatlah construct function di class controller
3. Buat lah method index, view,edit,simpan di class controller
4. Tampilkan sebuah text pada setiap method yang di buat ketika class method di aksess.

CHAPTER 6 : VIEWS

Apa itu Views?

View adalah kode yang bertugas untuk membuat tampilan aplikasi. View berisi kode campuran dari PHP, HTML, JS, dan CSS.

Ingat :

Fokusnya untuk membuat tampilan aplikasi, bukan yang lain. Kita tidak boleh query data dari view, meskipun itu bisa dilakukan.

Kadang juga kita akan membuat sedikit logika di dalam view, seperti logika untuk menampilkan dan menghilangkan elemen tertentu.

Apa yang kita lihat di aplikasi, itu adalah kode dari View.

Membuat Views

View dibuat di dalam folder application/views/ dan ada beberapa aturan penulisan view yang harus diperhatikan:

1. Penulisan nama file

   Nama file menggunakan huruf kecil dan jika terdiri dari dua suku kata, bisa dipisah dengan *undescore*.

   perhatikan contoh berikut:

   Contoh yang benar :

   welcome_message.php
   about.php
   overview.php

   Contoh yang salah :

   WelcomeMessage.php
   ABOUT.php
   over View.php

   Dalam penulisan nama file sebaiknya konsisten menggunakan huruf kecil dan *undescore*, jangan yang aneh-aneh agar kode lebih rapi.

2. Penulisan kode

File view bisa berisi kode PHP, HTML, CSS, dan JS. Karena tugasnya view untuk menampilkan output.. maka kita akan banyak menggunakan echo di sini.

Sangat tidak dianjurkan melakukan query data di sini, meskipun itu bisa dilakukan

Cara Load Views

Untuk memuat file tampilan tertentu, sebaiknya menggunakan metode berikut:

```
$this->load->view('name');
```

Di mana name adalah nama file tampilan yang telah dibuat.

Ekstensi file .php tidak perlu ditentukan kecuali Anda menggunakan sesuatu selain .php.

Sekarang, buka file controllers yang Anda buat sebelumnya bernama Blog.php, dan ganti pernyataan **echo** dengan metode load view:

perhatikan contoh berikut :

```
<?php
class Blog extends CI_Controller {


	public function index()
	{
		echo 'Hello World!'; //before
		$this->load->view('blogview'); // after
	}
}
```

Jika kita akses URI belajarcoding.com/index.php/blog secara otomatis file blogview akan terload dan akan tampil di halaman browser kita.

Load Banyak File Views Dalam Controllers

Dalam bebarapa kasus sering kali kita lihat banyak file views yang di buat dan di load secara bersamaan di controller pada method, pertanyaannya bagaimana cara nya me-load banyak file views dalam satu method?

Berikut penjelasaannya :

CodeIgniter akan dengan cerdas menangani beberapa panggilan ke $this->load->view() dari dalam pengontrol. Jika lebih dari satu panggilan terjadi mereka akan ditambahkan bersama-sama. Misalnya, Anda mungkin ingin memiliki tampilan header, tampilan menu, tampilan konten, dan tampilan footer. Itu mungkin terlihat seperti ini:

```
<?php
class Page extends CI_Controller {

	public function index()
	{
		$data['page_title'] = 'Your title';
		$this->load->view('header');
		$this->load->view('menu');
		$this->load->view('content', $data);
		$this->load->view('footer');
	}

}
```

Dalam contoh di atas, kita menggunakan "data yang ditambahkan secara dinamis" sebagai variable untuk menyimpan sebuat data yang akan di kirim ke file views. Data yang dikirim ke View berupa array asosiatif.

Menyimpan tampilan dalam Sub-direktori

File tampilan Anda juga dapat disimpan dalam sub-direktori jika Anda lebih suka jenis organisasi itu. Saat melakukannya, Anda harus menyertakan nama direktori yang memuat tampilan. Contoh:

```
$this->load->view('directory_name/file_name');
```

Menambahkan data Dinamis ke Views

Data diteruskan dari controllers ke views melalui array atau objek dalam parameter kedua dari metode load view. Berikut adalah contoh menggunakan array:

```
$data = array(
        'title' => 'My Title',
        'heading' => 'My Heading',
        'message' => 'My Message'
);

$this->load->view('blogview', $data);
```

Dan inilah contoh menggunakan objek:

```
$data = new Someclass();
$this->load->view('blogview', $data);
```

Contoh penggunaan:

Mari kita coba dengan file pengontrol Anda. Buka, tambahkan kode ini:

```
<?php
class Blog extends CI_Controller {

        public function index()
        {
                $data['title'] = "My Real Title";
                $data['heading'] = "My Real Heading";

                $this->load->view('blogview', $data);
        }
}
```

Sekarang buka file tampilan Anda dan ubah teks menjadi variabel yang sesuai dengan kunci array dalam data Anda:

```
<html>
<head>
<title><?php echo $title;?></title>
</head>
<body>
<h1><?php echo $heading;?></h1>
</body>
</html>
```

Kemudian muat halaman di URL yang telah Anda gunakan dan Anda akan melihat variabel diganti.

Cara Menampilkan Data di View

Saat kita melakukan load view dengan data seperti ini:

```
$data['name'] = 'Petani Kode';
$this->load->view('product.php', $data);
```

Kita bisa menampilkan isi datanya di dalam view dengan kode seperti ini:

```
<?php echo $name ?>
```

Variabel $name dapatnya dari mana?

Variabel $name didapatkan dari $data, perhatikanlah di sana ada key bernama name. Ini akan menjadi variabel di dalam view.

Sekarang mari kita pelajari lebih lanjut tentang cara menampilkan data.

1. **Shortcut untuk echo**

   Selain menggunakan fungsi echo, kita juga bisa memanfaatkan shortcut atau bentuk pendeknya dari echo.

   Perhatikan kode ini:

```
<?php echo $name ?>
```

Kita bisa menyingkatnya seperti ini:

```
<?= $name ?>
```

Kedua kode ini sebenarnya sama, hanya saja menggunakan <?= akan lebih pendek dibandingkan harus menulis <?php echo.

2. Percabangan

Pada controller kita load view dan mengirimkan data seperti ini:

```
$data['user'] ='petanikode';
$this->load->view('dashboard', $data);
```

Lalu pada view, kita bisa buat logika if/else seperti ini:

```
<?php
if ($user === 'petanikode'){
  echo "Welcome admin";
} else {
  echo "Hello guest";
}
?>
```

Atau bisa juga menggunakan operator ternary seperti ini:

```
<?= $user === 'admin' ? "Welcome admin" : "Hello guest"; ?>
```

Kadang kita juga akan menampilkan banyak kode HTML di dalam blok if/else.

Kita bisa saja melakukannya seperti ini:

```
<?php if ($user === 'petanikode'){ ?>
<h1>Hello Admin</h1>
```

```
<?php } else { ?>
<h1>Hello guest</h1?>
<?php } ?>
```

Ini boleh-boleh saja dilakukan, namun agak sulit terbaca karena harus mengingat tutup kurung }. Kadang-kadang ini membuat kita kesulitan.

Karena itu, gunakanlah bentuk if/else seperti ini:

```
<?php if ($user === "admin"): ?>
<h1>Hello Admin</h1>
<?php else: ?>
<h1>Hello guest</h1>
<?php endif ?>
```

Bentuk seperti ini lebih mudah dibaca karena kita tidak menggunakan kurung {}, melainkan menggunakan endif untuk menutup blok percabangan *if/else*.

3. Perulangan

Sama seperti percabangan, kita juga bisa melakukan perulangan pada View. Perulangan biasanya digunakan untuk menampilkan banyak data dari array.

```
$data['names'] = [
  "Foo",
  "Bar",
  "Bob"
];
$this->load->view('name', $data);
```

Maka di View kita bisa menggunakan perulangan seperti ini:

```
<?php
for ($i = 0, $i < count($names); $i++)
{
  echo $names[$i];
```

```
}


?>
```

Bisa juga dengan perulangan while seperti ini:

```
<?php
$i = 0;
while ($i < count($names))
{
  echo $names[$i];
  $i++;
}


?>
```

atau dengan perulangan foreach seperti ini:

```
<?php


foreach ($names as $name)
{
  echo $name;
}


?>
```

dan yang paling saya rekomendasikan adalah menggunakan *foreach* tanpa kurung {}.

Contoh:

```
<ul>
<?php foreach ($names as $name): ?>
<li><?= $name ?><li>
```

```
<?php endforeach ?>
</ul>
```

Bentuknya lebih bersih dan mudah dibaca.

Selain foreach, kita juga bisa menggunakan *for* dan *while* tanpa kurung {}.

Contoh:

```
<ul>
<?php for ($i = 0, $i < count($names); $i++): ?>
<li><?= $name ?><li>
<?php endfor ?>
</ul>
```

dan conoth while:

```
<ul>
<?php $i = 0; ?>
<?php while ($i < count($names)): ?>
<li><?= $name ?><li>
<?php endwhile ?>
</ul>
```

Latihan

1. Buat lah view dengan nama biodata.
2. Buat kerangka html data diri anda
3. Buat lah data diri anda di parshing dari controller

CHAPTER 7 : MODEL DAN DATABASE

Apa itu models?

Model adalah kelas PHP yang dirancang untuk bekerja dengan informasi dalam database Anda. Misalnya, Anda menggunakan CodeIgniter untuk mengelola blog. Anda mungkin memiliki kelas model yang berisi fungsi untuk menyisipkan, memperbarui, dan mengambil data blog Anda. Model bisa mengakses data dari Database dan juga sumber lainnya. Seperti API pihak ketiga.

Cara Membuat Models dan Aturan Pembuatan

Model dapat kita buat di dalam folder models, aturan penulisan model hampir sama dengan controller.

1. Penulisan nama

   Nama file model harus menggunakan huruf besar atau kapital di awal dan jika terdiri dari dua suku kata atau lebih, bisa dipisah dengan *underscore*.

   contoh:

   ```
   Arcticle_model.php
   Product_model.php
   Account_model.php
   ```

   Penamaan file model dengan akhiran _model boleh dilakukan boleh juga tidak, tujuan memberikan akhiran _model adalah untuk menghindari bentrok dengan nama Controller.

2. Penulisan nama class

   Class model harus melakukan *extends* dari class CI_Model.

   Nama class mengukuti nama file, yakni diawali dengan huruf besar dan boleh dipisah dengan *underscore*.

   contoh:

   ```
   <?php
   class Product_model extends CI_Model
   {
   ```

```
}
```

3. Penulisan nama method

   Nama method diawali dengan huruf kecil dan jika terdiri dari dua suku kata, boleh dipisah dengan *underscore*.

   contoh:

```
<?php

class Product_model extends CI_Model
{
    public function find()
  {

  }
  public function get_expired_product()
  {

  }
}
```

Cara Konfigurasi Database

CodeIgniter memiliki file konfigurasi yang memungkinkan Anda menyimpan nilai koneksi database Anda (nama pengguna, kata sandi, nama database, dll.). File konfigurasi terletak di application/config/database.php. Anda juga dapat mengatur nilai koneksi database untuk lingkungan tertentu dengan menempatkan database.php di folder konfigurasi lingkungan masing-masing.

Pengaturan konfigurasi disimpan dalam array multi-dimensi dengan prototipe ini:

```
$db['default'] = array(
        'dsn'   => '',
        'hostname' => 'localhost',
        'username' => 'root',
        'password' => '',
        'database' => 'database_name',
        'dbdriver' => 'mysqli',
        'dbprefix' => '',
        'pconnect' => TRUE,
        'db_debug' => TRUE,
        'cache_on' => FALSE,
        'cachedir' => '',
        'char_set' => 'utf8',
        'dbcollat' => 'utf8_general_ci',
        'swap_pre' => '',
        'encrypt' => FALSE,
        'compress' => FALSE,
        'stricton' => FALSE,
        'failover' => array()
)
```

Cara Koneksi ke Database

Ketika sebuah model dimuat itu TIDAK terhubung secara otomatis ke database Anda. Opsi berikut untuk menghubungkan tersedia untuk Anda:

- Anda dapat terhubung menggunakan metode database standar yang dijelaskan di sini, baik dari dalam kelas Controller atau kelas Model Anda.
- Anda dapat memberi tahu metode pemuatan model untuk terhubung secara otomatis dengan meneruskan TRUE (boolean) melalui parameter ketiga, dan pengaturan konektivitas, seperti yang ditentukan dalam file konfigurasi database Anda akan digunakan:

```
$this->load->model('model_name', '', TRUE);
```

- Anda dapat secara manual melewati pengaturan konektivitas database melalui parameter ketiga:

```
$config['hostname'] = 'localhost';
$config['username'] = 'myusername';
$config['password'] = 'mypassword';
$config['database'] = 'mydatabase';
$config['dbdriver'] = 'mysqli';
$config['dbprefix'] = '';
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;

$this->load->model('model_name', '', $config);
```

Query Pada Models

Di dalam model, kita harus menulis query untuk mengambil data di database dan mengembalikan hasil query berupa data. Nilai kembalian boleh berupa array maupun objek.

Contoh :

```
<?php

class Product_model extends CI_Model
{
   public function find()
  {
        $query = $this->db->get_where('table_name', ['id' => $id]);
        return $query->row(); // return berupa satu object
  }
```

```
 public function get_expired_product()
 {
        $query = $this->db->query("SELECT * FROM table WHERE expired=1");
        return $query->result(); // return berupa array objek
 }
}
```

Selain itu, ada juga yang menulis logika bisnis di dalam model.

contoh:

```
<?php

class Account_model extends CI_Model
{
 public function deposit($deposit_count, $id)
 {
        $current_balance = $this->db->get_where('balance', ['id' => $id]);
        $new_balance = $current_balance + $deposit_count;
        $this->db->update($new_balance, ['id' => $id]);
 }

 public function withdraw($wd_count, $id)
 {
        $current_balance = $this->db->get_where('balance', ['id' => $id]);
        $new_balance = $current_balance - $wd_count;
        $this->db->update($new_balance, ['id' => $id]);
 }
}
```

Pada contoh ini model menerima input dari argumen yang diberikan melalui method seperti $deposit_count, $wd_count, dan $id.

Lalu di simpan dengan query update().

Dari mana asalnya query update()?

Query ini berasal dari library Query Builder di Codeigniter.

Apa itu Query Build?

Query Builder adalah class yang berisi method untuk membuat query database.

Contoh:

- $this->db->insert() untuk menambahkan data ke tabel;
- $this->db->get() untuk mengambil data dari tabel, sama seperti query SELECT.
- $this->db->update() untuk update data di tabel;
- $this->db->delete() untuk menghapus data di tabel;
- $this->db->get_where() untuk mengambil data dengan WHERE;

Selain method-method ini, masih banyak lagi method lain yang disediakan Query Builder untuk membuat query database.

CHAPTER 8 : DATABASE QUERIES PREFERENCE

Mengenal Query Builder Class

CodeIgniter memberi Anda akses ke kelas Query Builder. Pola ini memungkinkan informasi untuk diambil, dimasukkan, dan diperbarui dalam database Anda dengan skrip minimal. Dalam beberapa kasus hanya satu atau dua baris kode yang diperlukan untuk melakukan tindakan database. CodeIgniter tidak mengharuskan setiap tabel database menjadi file kelasnya sendiri. Ini malah menyediakan antarmuka yang lebih disederhanakan.

Di luar kesederhanaan, manfaat utama menggunakan fitur Query Builder adalah memungkinkan Anda membuat aplikasi database independen, karena sintaks kueri dihasilkan oleh setiap adaptor database. Ini juga memungkinkan kueri yang lebih aman, karena nilai diloloskan secara otomatis oleh sistem.

Berikut list query class buildr yang akan di pelajari:

- Selecting Data
- Looking for Specific Data
- Inserting Data
- Updating Data
- Deleting Data

Selecting Data

Fungsi berikut memungkinkan Anda membuat pernyataan SQL SELECT.

```
$this->db->get()
```

Menjalankan kueri pemilihan dan mengembalikan hasilnya. Dapat digunakan dengan sendirinya untuk mengambil semua catatan dari tabel:

```
$query = $this ->db ->get ( 'mytable' );  // Menghasilkan: SELECT * FROM mytable
```

Parameter kedua dan ketiga memungkinkan Anda untuk menetapkan klausa batas dan offset:

```
$query = $this ->db ->get ( 'mytable' , 10 , 20 );

// Menjalankan: SELECT * FROM mytable LIMIT 20, 10

// (di MySQL. Database lain memiliki sintaks yang sedikit berbeda)
```

Anda akan melihat bahwa fungsi di atas ditetapkan ke variabel bernama $query, yang dapat digunakan untuk menampilkan hasil

$this->db->get_where()

Identik dengan fungsi di atas kecuali bahwa itu memungkinkan Anda untuk menambahkan klausa "where" di parameter kedua, alih-alih menggunakan fungsi db->where() :

contoh penggunaan:

```
$query = $this ->db ->get_where ( 'mytable' , array ( 'id' =>$id ), $limit , $offset );
```

$this->db->select()

Mengizinkan Anda untuk menulis bagian SELECT dari kueri Anda:

```
$this ->db ->pilih ( 'judul, konten, tanggal' );

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



// Jalankan: PILIH judul, konten, tanggal FROM mytable
```

Jika Anda memilih semua (*) dari tabel, Anda tidak perlu menggunakan fungsi ini. Ketika dihilangkan, CodeIgniter mengasumsikan bahwa Anda ingin memilih semua bidang dan secara otomatis menambahkan 'select *'.

$this->db->select() menerima parameter kedua opsional. Jika Anda menyetelnya ke FALSE, CodeIgniter tidak akan mencoba melindungi nama bidang atau tabel Anda. Ini berguna jika Anda memerlukan pernyataan pemilihan majemuk di mana pelepasan bidang secara otomatis dapat merusaknya.

Contoh

```
$this ->db ->pilih ( '(SELECT SUM(payments.amount) FROM payments WHERE payments.invoice_id=4) AS number_paid' , FALSE );

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

Beberapa jenis select pada query class select();

$this->db->select_max()

Menulis sebagian untuk kueri Anda. Anda dapat secara opsional menyertakan parameter kedua untuk mengganti nama bidang yang dihasilkan.SELECT MAX(field)

contoh:

```
$this ->db ->select_max ( 'umur' );
$query = $this ->db ->get ( 'members' );  // Menghasilkan: SELECT MAX(umur) sebagai umur DARI anggota

$this ->db ->select_max ( 'umur' , 'member_age' );
$query = $this ->db ->get ( 'members' ); // Menghasilkan: PILIH MAX(umur) sebagai member_age FROM member
```

$this->db->select_min()

Menulis bagian “PILIH MIN(bidang)” untuk kueri Anda. Seperti halnya select_max(), Anda dapat secara opsional menyertakan parameter kedua untuk mengganti nama bidang yang dihasilkan.

contoh:

```
$this ->db ->select_min ( 'umur' );

$query = $this ->db ->get ( 'members' ); // Menghasilkan: PILIH MIN(umur) sebagai umur DARI anggota
```

$this->db->select_avg()

Menulis bagian “PILIH AVG(bidang)” untuk kueri Anda. Seperti halnya select_max(), Anda dapat secara opsional menyertakan parameter kedua untuk mengganti nama bidang yang dihasilkan.

contoh:

```
$this ->db ->select_avg ( 'umur' );

$query = $this ->db ->get ( 'members' ); // Menghasilkan: PILIH AVG(usia)
```

```
sebagai usia DARI anggota
```

$this->db->select_sum()

Menulis bagian “SELECT SUM(field)” untuk kueri Anda. Seperti halnya select_max(), Anda dapat secara opsional menyertakan parameter kedua untuk mengganti nama bidang yang dihasilkan.

contoh:

```
$this ->db ->select_sum ( 'umur' );

$query = $this ->db ->get ( 'members' ); // Menghasilkan: SELECT SUM(umur) sebagai umur DARI anggota
```

$this->db->from()

Mengizinkan Anda menulis bagian FROM dari kueri Anda:

```
$this ->db ->pilih ( 'judul, konten, tanggal' );

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

$query = $this ->db ->get (); // Menghasilkan: PILIH judul, konten, tanggal FROM mytable
```

$ini->db->join()

Mengizinkan Anda untuk menulis bagian GABUNG dari kueri Anda:

```
$this ->db ->pilih ( '*' );

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

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

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



// Menghasilkan:

// SELECT * FROM blogs JOIN comments ON comments.id = blogs.id
```

Looking for Specific Data

$this->db->where()

Fungsi ini memungkinkan Anda untuk menyetel klausa **WHERE** menggunakan salah satu dari empat metode:

Semua nilai yang diteruskan ke fungsi ini diloloskan secara otomatis, menghasilkan kueri yang lebih aman.

Metode kunci/nilai sederhana:

```
$this -> db -> where ( 'nama' , $nama ); // Menghasilkan: WHERE name = 'Joe'
```

Perhatikan bahwa tanda sama dengan ditambahkan untuk Anda.

Jika Anda menggunakan beberapa panggilan fungsi, mereka akan dirantai bersama-sama dengan DAN di antara mereka:

```
$this ->db ->where ( 'nama' , $nama );

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

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

// WHERE name = 'Joe' AND title = 'boss' AND status = 'active'
```

Metode kunci/nilai khusus:

Anda dapat menyertakan operator di parameter pertama untuk mengontrol perbandingan:

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

```
$this ->db ->where( 'id <' , $id ); // Menghasilkan: WHERE name != 'Joe' AND
id < 45
```

Metode array asosiatif:

```
$array = array ( 'name' =>$name , 'title' =>$title , 'status' =>$status );

$this->db->di mana ( $array );

// Menghasilkan: WHERE name = 'Joe' AND title = 'boss' AND status = 'active'
```

Anda dapat memasukkan operator Anda sendiri menggunakan metode ini juga:

```
$array = array ( 'name !=' =>$name , 'id <' =>$id , 'date >' =>$date );

$this ->db ->di mana ( $array );
```

String khusus:

Anda dapat menulis klausa Anda sendiri secara manual:

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

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

Inserting Data

$this->db->insert()

Menghasilkan string sisipan berdasarkan data yang Anda berikan, dan menjalankan kueri. Anda bisa melewatkan **array** atau **objek** ke fungsi. Berikut adalah contoh menggunakan array:

```
$data=array(

'title'=>'My title',

'name'=>'My Name',

'date'=>'My date'

);



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

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

Parameter pertama akan berisi nama tabel, yang kedua adalah array nilai asosiatif.

Berikut adalah contoh menggunakan objek:

```
/*

class Myclass {

        public $title = 'My Title';

        public $content = 'My Content';

        public $date = 'My Date';

}

*/



$object=new Myclass;

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

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

'My Content', 'My Date')
```

Parameter pertama akan berisi nama tabel, yang kedua adalah objek.

$this->db->insert_batch()

Menghasilkan string sisipan berdasarkan data yang Anda berikan, dan menjalankan kueri. Anda bisa melewatkan **array** atau **objek** ke fungsi. Berikut adalah contoh menggunakan array:

```
$data=array(

array(

'title'=>'My title',

'name'=>'My Name',

'date'=>'My date'

        ),

array(

'title'=>'Another title',

'name'=>'Another Name',

'date'=>'Another date'

        )

);



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

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

name', 'My date'),  ('Another title', 'Another name', 'Another date')
```

Parameter pertama akan berisi nama tabel, yang kedua adalah array nilai asosiatif.

Updating Data

$this->db->replace()

Metode ini mengeksekusi pernyataan REPLACE, yang pada dasarnya adalah standar SQL untuk (opsional) DELETE + INSERT, menggunakan kunci *PRIMARY* dan *UNIQUE* sebagai faktor penentu. Dalam kasus kami, ini akan menyelamatkan Anda dari kebutuhan untuk menerapkan logika kompleks dengan kombinasi yang berbeda dari select(), update(), delete()dan insert()panggilan.

Contoh

```
$data=array(

'title'=>'My title',

'name'=>'My Name',

'date'=>'My date'

);



$this->db->replace('table', $data);



// Executes: REPLACE INTO mytable (title, name, date) VALUES ('My title', 'My name', 'My date')
```

Dalam contoh di atas, jika kita menganggap bahwa bidang *judul* adalah kunci utama kita, maka jika baris yang berisi 'Judul saya' sebagai nilai *judul* , baris tersebut akan dihapus dengan data baris baru yang menggantikannya.

Penggunaan set()metode ini juga diperbolehkan dan semua bidang diloloskan secara otomatis, sama seperti dengan insert().

$this->db->set()

Fungsi ini memungkinkan Anda untuk menetapkan nilai untuk sisipan atau pembaruan.

Ini dapat digunakan daripada meneruskan array data langsung ke fungsi insert atau update:

```
$this->db->set('name', $name);

$this->db->insert('mytable');  // Produces: INSERT INTO mytable (`name`) VALUES ('{$name}')
```

Jika Anda menggunakan beberapa fungsi yang dipanggil, mereka akan dirakit dengan benar berdasarkan apakah Anda melakukan penyisipan atau pembaruan:

```
$this->db->set('name', $name);

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

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

$this->db->insert('mytable');
```

set() juga akan menerima parameter ketiga opsional ($escape), yang akan mencegah data diloloskan jika disetel ke FALSE. Untuk mengilustrasikan perbedaannya, di sini set()digunakan baik dengan dan tanpa parameter escape.

```
$this->db->set('field', 'field+1', FALSE);

$this->db->where('id', 2);

$this->db->update('mytable'); // gives UPDATE mytable SET field = field+1 WHERE id = 2



$this->db->set('field', 'field+1');

$this->db->where('id', 2);

$this->db->update('mytable'); // gives UPDATE `mytable` SET `field` = 'field+1' WHERE `id` = 2
```

Anda juga dapat meneruskan array asosiatif ke fungsi ini:

```
$array=array(

'name'=>$name,

'title'=>$title,

'status'=>$status

);



$this->db->set($array);

$this->db->insert('mytable');
```

Atau objek:

```
/*

class Myclass {

        public $title = 'My Title';

        public $content = 'My Content';

        public $date = 'My Date';

}

*/



$object=new Myclass;

$this->db->set($object);

$this->db->insert('mytable');
```

$this->db->update()

Menghasilkan string pembaruan dan menjalankan kueri berdasarkan data yang Anda berikan. Anda dapat melewatkan **array** atau **objek** ke fungsi. Berikut adalah contoh menggunakan array:

```
$data=array(

'title'=>$title,
```

```
'name'=>$name,

'date'=>$date

);



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

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

// Produces:

//

//      UPDATE mytable

//      SET title = '{$title}', name = '{$name}', date = '{$date}'

//      WHERE id = $id
```

Atau Anda dapat menggunakan objek:

```
/*

class Myclass {

        public $title = 'My Title';

        public $content = 'My Content';

        public $date = 'My Date';
```

```
}

*/



$object=new Myclass;

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

$this->db->update('mytable', $object);

// Produces:

//

// UPDATE `mytable`

// SET `title` = '{$title}', `name` = '{$name}', `date` = '{$date}'

// WHERE id = `$id`
```

$this->db->update_batch()

Menghasilkan string pembaruan berdasarkan data yang Anda berikan, dan menjalankan kueri. Anda bisa melewatkan **array** atau **objek** ke fungsi. Berikut adalah contoh menggunakan array:

```
$data=array(

array(

'title'=>'My title' ,
```

```
'name'=>'My Name 2' ,

'date'=>'My date 2'

  ),

array(

'title'=>'Another title' ,

'name'=>'Another Name 2' ,

'date'=>'Another date 2'

  )

);



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



// Produces:

// UPDATE `mytable` SET `name` = CASE

// WHEN `title` = 'My title' THEN 'My Name 2'

// WHEN `title` = 'Another title' THEN 'Another Name 2'

// ELSE `name` END,
```

```
// `date` = CASE

// WHEN `title` = 'My title' THEN 'My date 2'

// WHEN `title` = 'Another title' THEN 'Another date 2'

// ELSE `date` END

// WHERE `title` IN ('My title','Another title')
```

Deleting Data

Menghasilkan string SQL hapus dan menjalankan kueri.

```
$this->db->delete('mytable', array('id'=>$id));  // Produces: // DELETE FROM

mytable  // WHERE id = $id
```

Parameter pertama adalah nama tabel, yang kedua adalah klausa where. Anda juga dapat menggunakan fungsi where() atau or_where() alih-alih meneruskan data ke parameter kedua fungsi:

```
$this->db->where('id', $id);

$this->db->delete('mytable');



// Produces:

// DELETE FROM mytable

// WHERE id = $id
```

Array nama tabel dapat diteruskan ke delete() jika Anda ingin menghapus data dari lebih dari 1 tabel.

```
$tables=array('table1', 'table2', 'table3');

$this->db->where('id', '5');

$this->db->delete($tables);
```

MINI PROJECT 1 : FITUR LOGIN

Langkah 1 : Persiapan dan Konfigurasi Database

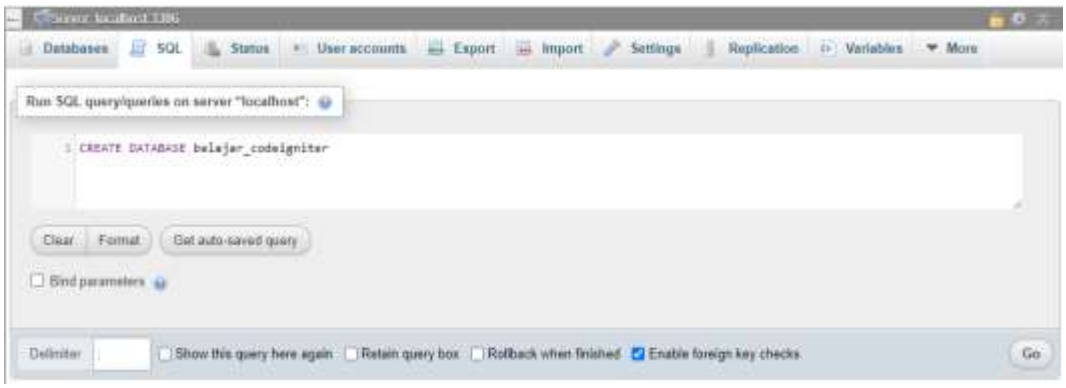
Buatlah database di phpmyadmin dengan cara akses localhost/phpmyadmin, buat database dengan nama **belajar_codeigniter.** kemudian buat table dengan nama **admin**. yang berisi 3 column yakni id,username dan password. silahkan copy and paste code dibawah ini.

Ikuti langkah - langkah berikut untuk pembuatan database.

Step 1 : Membuat database

```
CREATEDATABASEbelajar_codeigniter
```

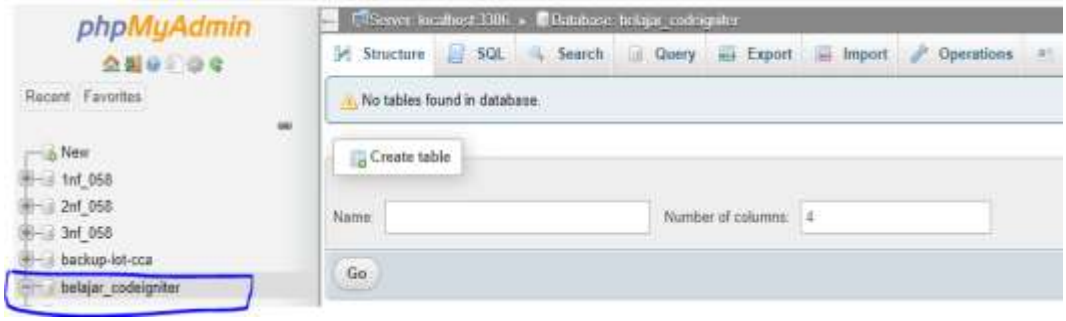
copy atau ketikan code di atas ke dalam editor sql di phpmyadmin.



Jika suda di copy atau di ketikan klik tombol go atau kirim. maka database berhasil di buat

Step 2 : membuat table di database

silahkan pilih database yang sudah di buat sebelumnya, kemudian kilk nama database lihat yang di lingkari warna biru.



Maka anda akan melihat tapilan form create table. silahkan pilih tab sql untuk membuka editor sql.

```
CREATETABLEIFNOTEXISTS`admin` (
`id`int(11) NOT NULL AUTO_INCREMENT,
`username`varchar(255) NOT NULL,
`password`varchar(255) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;

--
-- Dumping data for table `admin`
--

INSERT INTO`admin` (`id`, `username`, `password`) VALUES
(1, 'admin', '21232f297a57a5a743894a0e4a801fc3');
```

Copy atau ketikan code di atas ke dalam editor sql dan klik tombol go atau kirim. secara otomatis akan membuat sebut table dengan nama admin beserta kolom id, username dan password. seperti gambar berikut

Langkah 2 : Konfigurasi Project File

Untuk konfigurasi atau pengaturan file project lihat pada chapter 3 pada pont B.

Langkah 3 : Konfigurasi Awal CodeIgniter

pada studi kasus kali ini kita akan mengkonfigurasi codeigniter agar dapat di akses untuk keperluan kita. sepeti liblary, helper maupun koneksi ke database kita.

Note: pada studi kasus ini kita menggunakan md5 sebagai enkripsi password yang kita gunakan.

Step 1: mengaktifkan liblary

silahkan edit file **autoload.php** pada folder **application/config/autoload.php.**

edit code yang ada di autoload.php sesuai dengan code di bawah ini

```
$autoload['libraries'] = array('database','session');

$autoload['helper'] = array('url');
```

Step 2 : setting config.php

silahkan edit file **config.php** pada folder **application/config/config.php.**

edit code yang ada di config.php sesuai dengan code di bawah ini :

```
$config['base_url']  =  ((isset($_SERVER['HTTPS'])  &&$_SERVER['HTTPS'] == "on") ? "https" : "http");

$config['base_url'] .= "://" . $_SERVER['HTTP_HOST'];

$config['base_url']   .=   str_replace(basename($_SERVER['SCRIPT_NAME']), "", $_SERVER['SCRIPT_NAME']);


$config['encryption_key'] = 'login-codeigniter';
```

Step 3 : menghubungkan ke database

silahkan edit file **database.php** pada folder **application/config/database.php.**

edit code yang ada di database.php sesuai dengan code di bawah ini :

```
$db['default'] = array(

'dsn'   =>'',

'hostname' =>'localhost',

'username' =>'root',

'password' =>'',

'database' =>'belajar_codeigniter',

'dbdriver' =>'mysqli',

'dbprefix' =>'',

'pconnect' =>FALSE,
```

```
'db_debug' => (ENVIRONMENT !== 'production'),

'cache_on' =>FALSE,

'cachedir' =>'',

'char_set' =>'utf8',

'dbcollat' =>'utf8_general_ci',

'swap_pre' =>'',

'encrypt' =>FALSE,

'compress' =>FALSE,

'stricton' =>FALSE,

'failover' =>array(),

'save_queries' =>TRUE

);
```

Langkah 4 : Membuat Controller

Langkah selanjutnya buat sebuah controller dengan nama Login.php, di controller ini yang akan kita tugaskan untuk menampilkan form login dan melakukan proses verifikasi/authentikasi username dan password admin yang di masukan, serta juga membuat fungsi untuk logout.

Buat file baru dengan nama **Login.php** di folder application/controller. kemudian copy atau ketikan code di bawah ini ke dalam file Login.php

Login.php

```
<?php

classLoginextendsCI_Controller
```

```
{
function__construct()
    {
parent::__construct();
$this->load->model('m_login');
    }

functionindex()
    {
$this->load->view('v_login');
    }

functionaksi_login()
    {
$username = $this->input->post('username');
$password = $this->input->post('password');
$where = array(
'username' =>$username,
'password' =>md5($password)
        );
$cek = $this->m_login->cek_login("admin", $where)->num_rows();
if ($cek>0) {
```

```
$data_session = array(
'nama' =>$username,
'status' =>"login"
            );

$this->session->set_userdata($data_session);

redirect(base_url("admin"));
        } else {
echo"Username dan password salah !";
        }
    }

functionlogout()
    {
$this->session->sess_destroy();
redirect(base_url('login'));
    }
}
```

Buat controller admin dengan nama Admin.php dan simpan ke folder application/controllers.

copy atau ketikan code di bawah ini ke dalam file Admin.php.

Admin.php

```
<?php

classAdminextendsCI_Controller

{

function__construct()

   {

parent::__construct();

if ($this->session->userdata('status') != "login") {

redirect(base_url("login"));

      }

   }

functionindex()

   {

$this->load->view('v_admin');

   }

}
```

Langkah 5 : Membuat Model

buat lah file dengan nama M_login pada folder application/models.

copy atau ketikan code di bawah ini kedalam file M_login.php

M_login.php

```
<?php

classM_loginextendsCI_Model
{
functioncek_login($table, $where)
   {
return$this->db->get_where($table, $where);
   }
}
```

Langkah 6 : Membuat View

buat lah file baru dengan nama v_login.php simpan ke folder application/views

copy atau ketikan code di bawah ini kedalam file v_login.php

v_login.php

```
<!DOCTYPEhtml>
<html>
<head>
```

```
<title>Membuat Login Dengan CodeIgniter</title>
</head>
<body>
<h1>Membuat Login Dengan CodeIgniter
<formaction="<?phpechobase_url('index.php/login/aksi_login');
?>"method="post">
<table>
<tr>
<td>Username</td>
<td><inputtype="text"name="username"></td>
</tr>
<tr>
<td>Password</td>
<td><inputtype="password"name="password"></td>
</tr>
<tr>
<td></td>
<td><inputtype="submit"value="Login"></td>
</tr>
</table>
</form>
</body>
```

```
</html>
```

Buat file view admin dengan nama v_admin.php dan simpan ke folder application/views

copy atau ketikan code berikut kedalam file v_admin.php

v_admin.php

```
<!DOCTYPEhtml>
<html>
<head>
<title>Membuat login dengan codeigniter</title>
</head>
<body>
<h1>Login berhasil !</h1>
<h2>Hai, <?phpecho$this->session->userdata("nama"); ?></h2>
<ahref="<?phpechobase_url('login/logout'); ?>">Logout</a>
</body>
</html>
```

Langkah 7 : Pengujian

Buka browser chrome atau firefox kemudian akses url berikut :

“belajarcodeigniter.test/index.php/login” di browser anda

Ketikan admin dikolom username dan ketikan admin di kolom password. kemudian klik tombol login, jika login berhasil akan tampil halaman admin seperti berikut :

Jika login salah akan tampil halaman berikut :

DAFTAR PUSTAKA

Alelo, F. M., Kmurawak, R. M., & Sampebua, M. R. (2021). Sistem Informasi Setoran Wajib Jemaat Menggunakan Framework Codeigniter. *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, *10*(2), 224–231. https://doi.org/10.32736/sisfokom.v10i2.1143

Bagus Wibisono, S. (2020). *Pengembangan Website Menggunakan Framework Codeigniter Sebagai Penunjang Sistem SMS Gateway di Iphone Bali*.

Chusyairi, A., Setiyadi, D., Saludin, S., & Rusmawan, U. (2020). PKM Pengenalan Online PHP dengan CI untuk ASN dan Non ASN Pemerintah Kota Bekasi. *CARADDE: Jurnal Pengabdian Kepada Masyarakat*, *3*(1), 166–173.

Fadilah, S. C., Rianto, H., & Hartati, T. (2020). Implementasi Framework Code Iginter Menggunakan Metode Waterfall Pada Sistem Informasi Penjualan Pt . Supreme Jaya Abadi Jisicom. *Journal Of Information System, Informatics and Computing*, *4*(1), 134–140.

Fadllullah, A., Rudy, & Mahdi, S. (2022). Rancang Bangun Simdalev Berbasis Framework CodeIgniter-Harviacode untuk Manajemen Pengendalian dan Evaluasi Pembangunan Daerah Kabupaten Tana Tidung. *Jurnal Teknik Komputer AMIK BSI*, *8*(2), 174–180. https://doi.org/10.31294/jtk.v4i2

Hapsari, I. N. (2020). Penyuluhan Mengenai Pengaturan Laravel Pada Windows Bagi Siswa Smk Yayasan Cinta Kasih Tzu Chi. *Jurnal Abdimas*, *6*(2), 82–87.

Irawan, P., Sokibi, P., & Prasetya Dimas Aulia Pudjie. (2020). Rancang Bangun Sistem Pengarsipan Surat Kedinasan. *Jurnal Manajemen Informatika & SIstem Informasi*, *3*(2), 157–165.

Rahman, A. (2023). *Rancang Bangun Website Sekolah Dengan Menggunakan Framework Codeigniter 3 (Studi Kasus : SDN 12 OKU)*. *19*(1), 162–167.

Rasikhah, H., & Adriansyah, A. R. (2022). Perancangan dan Implementasi Booking System Lapangan menggunakan Framework MVC berbasis Web. *Jurnal Informatika Terpadu*, *8*(1), 08–12. https://doi.org/10.54914/jit.v8i1.384

Ridwan, M., Sinaga, T. H., & Elsera, M. (2022). *PENERAPAN FRAMEWORK CODEIGNITER DALAM PERANCANGAN*. *3*(1).

Syafitri, Y., Pramudya, Y. D., & Rasid, M. (2021). Pemanfaatan Framework Codeigniter Untuk Membangun Aplikasi Display Produk Di Alfamart Rajabasa. *Jurnal Informasi dan Komputer*, *9*(1), 45–52. https://doi.org/10.35959/jik.v9i1.205

ISBN 978-623-5745-78-7 (PDF)

CV. KIREINARA
Publishing, Distributting & Training