



Tutorial Dasar Laravel

Teten Nugraha – tennugraha777@gmail.com

DAFTAR ISI

1. Berkenalan dengan Laravel
 - 1.1. Apa itu Laravel ?
 - 1.2. Kenapa Memakai Laravel ?
2. Memulai Laravel (Instalasi dan Konfigurasi)
 - 2.1 Requirement
 - 2.2 Install Package Laravel
 - 2.3 Struktur Projek Laravel
3. Mengenal Dasar Routing
 - 3.1 Routing Dasar
 - 3.2 Routing Berparameter
4. Mengenal MVC (Model-View-Controller)
5. Mengenal Blade Template Laravel
6. Form dan HTML
7. Schema Builder
8. Migrations
9. Seeding
10. Eloquent

Koreksi

Sebenarnya saya masih belajar framework laravel ini, disamping saya belajar saya juga didokumentasikan hasil belajar itu menjadi *ebook* ini. Jika teman-teman menemukan kesalahan penjelasan yang ditulis oleh Saya silahkan kirim komentar nya ke email tennugraha777@gmail.com.

Untuk bab yang lengkap nya nanti saya bagikan kembali.

Mudah-mudahan *ebook* ini sedikitnya dapat membantu teman-teman yang akan belajar framework laravel ini.

Terima Kasih

Teten Nugraha

1. Berkenalan dengan Laravel



1.1. Apa itu Laravel ?

Laravel adalah sebuah Framework PHP 5.3 yang bersifat *opensource* yang ditulis oleh Taylor Otwell dengan lisensi dibawah *MIT License*. Laravel dibuat untuk membantu para developer khususnya dalam membuat sebuah web dengan sintak yang sederhana, elegan, ekspresif dan menyenangkan. Seperti yang ditulis dalam website nya :

Laravel is a clean and classy framework for PHP web development. Freeing you from spaghetti code, it helps you create wonderful applications, using simple, expressive syntax. Development should be a creative experience that you enjoy, not something that is painful. Enjoy the fresh air!

Laravel adalah aplikasi web dengan sintak yang ekspresif dan elegan. Dengan Laravel, tugas-tugas umum developer dapat dikurangi pada sebagian besar proyek-proyek web seperti *routing*, *session* dan *caching*. Disamping itu, laravel berusaha menggabungkan pengalaman-pengalaman development dalam bahasa lain, seperti Ruby on Rails, ASP.NET, MVC dan Sinatra.

1.2. Kenapa Memakai Laravel ?

Sesuai dengan motto laravel itu sendiri "***PHP doesn't hurt, code happy & enjoy the fresh air***". Tujuan utama dari laravel adalah mempermudah *coding* dalam membuat sebuah produk web. Bahkan laravel termasuk dalam **best php framework 2014** versi **webdesignmoo** dan yang paling banyak digunakan oleh developer. Ini membuktikan bahwa menggunakan Laravel memang dapat mempercepat dan mempermudah *development* website.

2. Memulai Laravel (Instalasi dan Konfigurasi)

Laravel sangatlah mudah untuk dikonfigurasi untuk mengembangkan sebuah proyek. Pada bagian ini, saya akan menjelaskan software/tools apa saja yang diperlukan, proses instalasi dan proses konfigurasinya.

2.1. Requirement

Adapun kebutuhan yang harus disediakan diantaranya :

1. Text Editor

Pilih text editor yang sesuai dengan kebutuhan atau selera Anda. Penulis menggunakan text editor Sublime Text Versi 3 yang sudah diinstal Emmet (plugin untuk mempercepat penulisan kode HTML). Bisa juga menggunakan PHPStorm, Aptana, Netbeans, Notepad++ dan lain-lain.

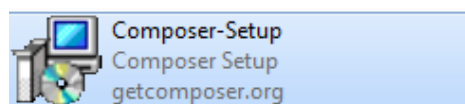
2. Webserver dan Database

Yang terpenting dalam instalasi Laravel yaitu bahwa persi PHP minimal versi 5.3 ke atas dan sudah diinstal ekstensi php yaitu Mcrypt. Penulis menggunakan XAMPP versi 3.2.1 yang sudah mendukung php5.4 dan Mcrypt. Dapat anda unduh di website nya atau sudah disediakan dalam paket CD.

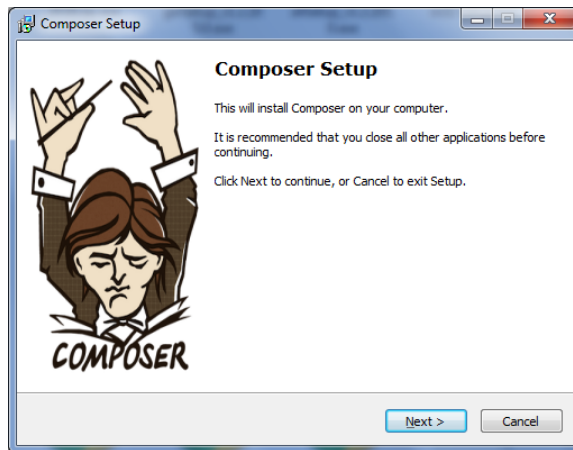
3. Composer

Untuk dapat menginstal laravel kita akan menggunakan composer. Composer adalah sebuah *dependency* 'manager' untuk PHP. Anda dapat menambah library yang dibutuhkan untuk website Anda secara otomatis tanpa perlu mendownload satu persatu. Mirip dengan apt-get install pada sistem operasi linux. Adapun proses instalasi composer adalah sebagai berikut :

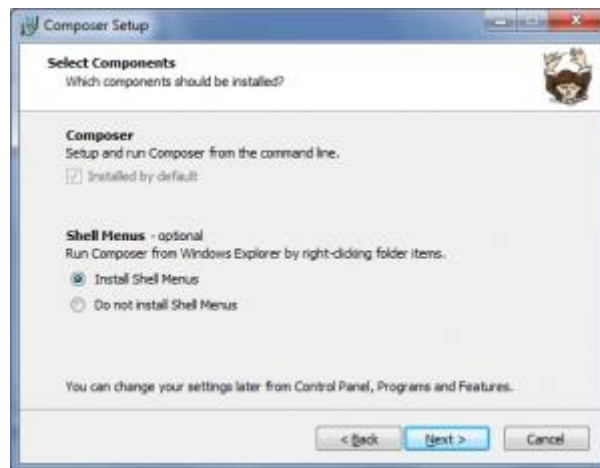
- Unduh composer di <https://getcomposer.org/Composer-Setup.exe>,



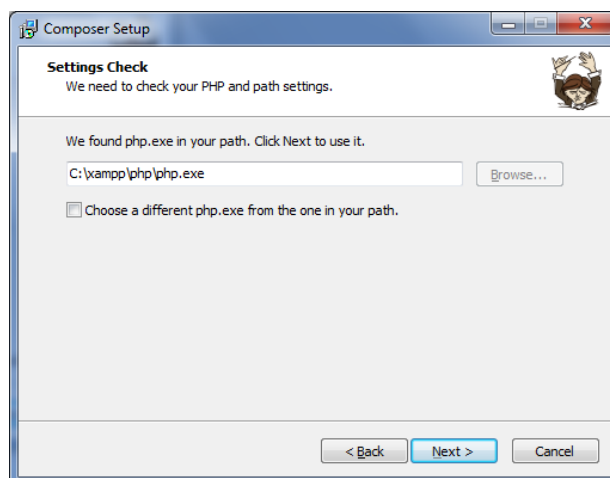
- Klik dua kali file tersebut sehingga muncul dialog setup install composer kemudian klik Next



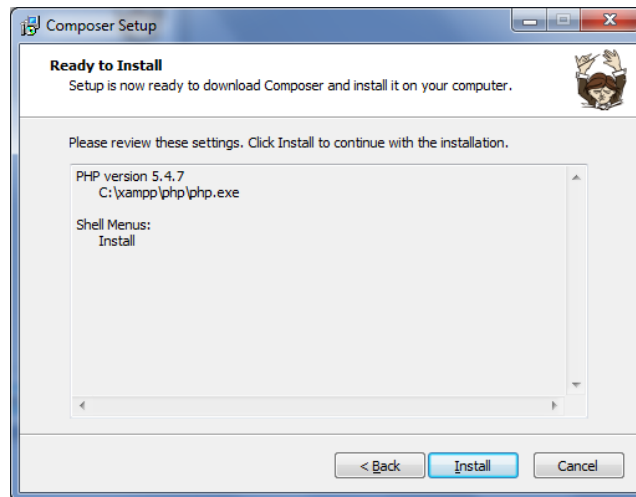
- Memilih komponen yang akan diinstall. Pilih “Install Shell Menus” kemudian klik next



- Check versi php. Pada tombol “browse”, kemudian masukan path php yang sudah diinstall di komputer. Disini dicontohkan path php nya yaitu di “C:/xampp/php/php.exe” kemudian klik next.



- Jika versi php sudah memenuhi standar instalasi Laravel, maka akan muncul tampilan sebagai berikut. Kemudian klik Install



- Jika sudah berhasil instal composer, untuk mengecek apakah composer sudah berjalan dengan baik, buka Command Prompt kemudian tuliskan “*composer -v*”. Jika berhasil composer akan tampil sebagai berikut .

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\admin>composer -v

Composer version e707dcd92f63236124ddb00eb625f53a2299bf07 2014-05-07 17:29:59

Usage:
  [options] command [arguments]

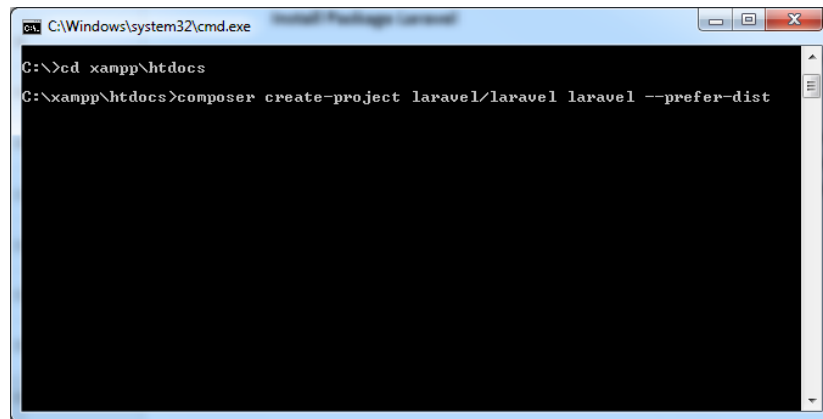
Options:
--help             -h Display this help message.
--quiet            -q Do not output any message.
--verbose          -v|vv|vvv Increase the verbosity of messages: 1 for normal ou
tput, 2 for more verbose output and 3 for debug
--version          -V Display this application version.
--ansi             Force ANSI output.
--no-ansi          Disable ANSI output.
--no-interaction   -n Do not ask any interactive question.
--profile          Display timing and memory usage information
  
```

2.2. Install Package Laravel

1. Untuk mengunduh paket laravel dapat menggunakan dua cara yaitu :
 - a. Mendownload paket laravel dari github di <https://github.com/laravel/laravel/archive/master.zip>
 - b. Atau mendownload langsung menggunakan composer langsung (penulis disini menggunakan composer langsung untuk proses download nya).
2. Buka Command Prompt kemudian arahkan direktori ke htdoc XAMP yang sudah di instal .kemudian ketikan sintak berikut :

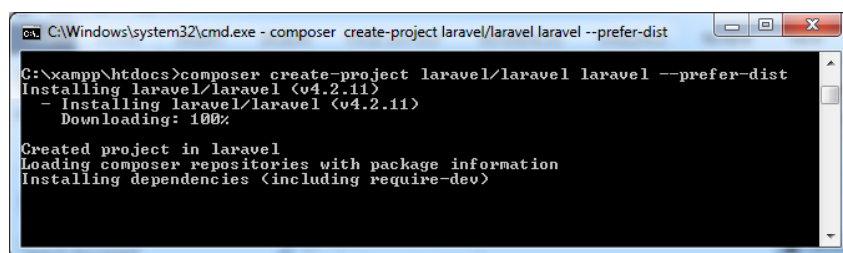
composer create-project laravel/laravel laravel --prefer-dist <Enter>

sintak diatas untuk mengunduh laravel dengan nama projek nya “laravel”



```
C:\Windows\system32\cmd.exe
C:\>cd xampp\htdocs
C:\xampp\htdocs>composer create-project laravel/laravel laravel --prefer-dist
```

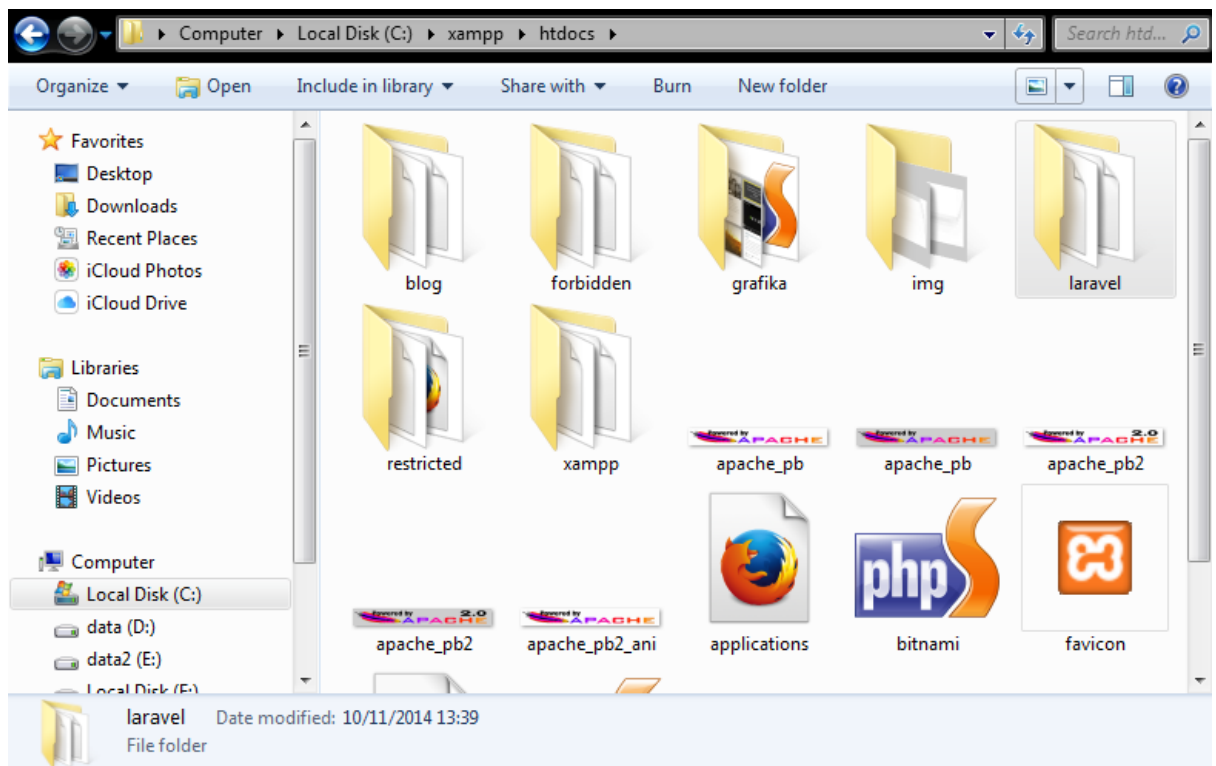
3. Tunggu sampai proses unduh berhasil



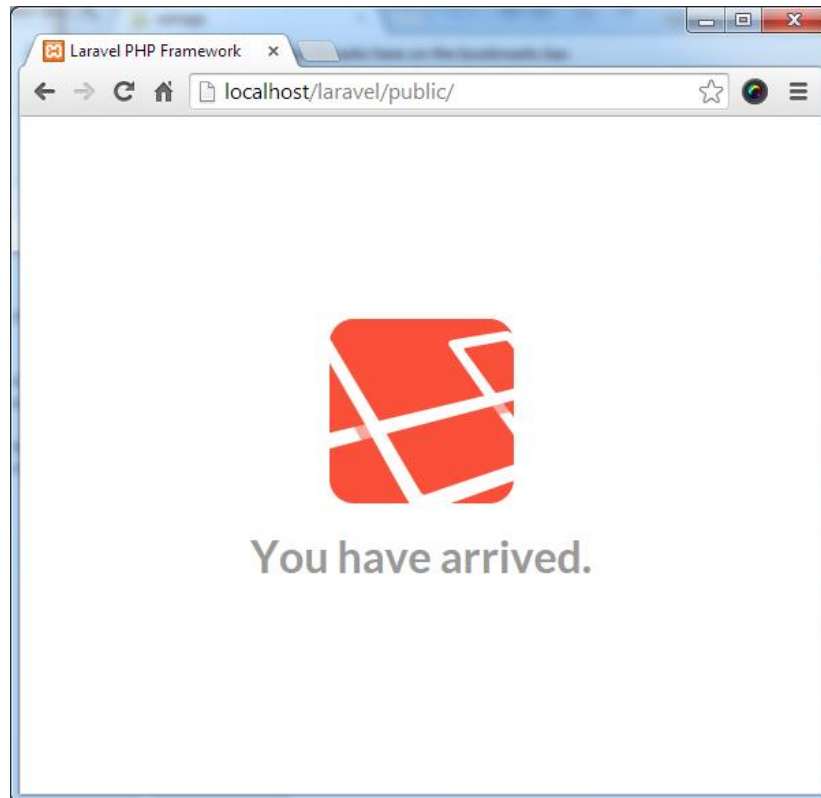
```
C:\Windows\system32\cmd.exe - composer create-project laravel/laravel laravel --prefer-dist
C:\xampp\htdocs>composer create-project laravel/laravel laravel --prefer-dist
Installing laravel/laravel (v4.2.11)
- Installing laravel/laravel (v4.2.11)
  Downloading: 100%

Created project in laravel
Loading composer repositories with package information
Installing dependencies (including require-dev)
```

4. Jika proses unduh telah selesai, kemudian buka explorer>htdoc akan ada file laravel yang sudah didownload tadi.



5. Untuk mengecek apakah laravel sudah terinstal yaitu dengan cara buka xampp-control kemudian centang apache dan mysql. Buka browser kemudian ketik <http://localhost/laravel/public> . Jika tampilan sebagai berikut, berarti paket laravel yang diinstal sudah berjalan.



2.3. Struktur Proyek Laravel

Jika kita buka folder laravel tersebut maka kita akan menemukan folder-folder dan file sebagai berikut :

- app/
- bootstrap/
- vendor/
- public/
- .gitattributes
- .gitignore
- arisan
- composer.json
- composer.lock
- phpunit.xml
- server.php

berikut adalah penjelasan struktur-struktur proyek laravel diatas .

- **app/**
folder ini digunakan untuk menyediakan tempat default untuk menyimpan kode yang sudah ditulis atau dikonfigurasi. Pada folder ini lah kita meletakkan semua kode proyek aplikasi yang dimulai dari konfigurasi, logic dan sebagainya. Didalam folder **app** juga terdapat beberapa folder dan file yang akan dijelaskan dibawah ini.

- ✓ **commands/**, folder ini merupakan folder yang digunakan untuk menyimpan perintah-perintah arisan yang dibuat oleh anda sendiri untuk keperluan aplikasi Anda.
- ✓ **config/**, folder ini merupakan folder yang berisi tentang konfigurasi baik untuk framework ataupun aplikasi anda. Selain itu anda juga dapat membuat folder sesuai dengan keinginan anda misalnya untuk menampung kelas-kelas validasi buatan Anda.
- ✓ **controllers/**, folder ini digunakan untuk menyimpan kelas-kelas PHP controller Anda. Dengan menggunakan controller maka anda dapat memisahkan logika aplikasi anda dalam beberapa kelas PHP.
- ✓ **databases/**, folder ini merupakan tempat untuk menyimpan keperluan basisdata baik untuk migrasi (*migration*) maupun untuk memasukan data ke basisdata (*seeds*).
- ✓ **lang/**, folder ini digunakan untuk menyimpan berbagai lokalisasi bagasa utuk keperluan pengembangan aplikasi, paginasi, validasi dan lainnya.
- ✓ **models/**, folder yang berisi kelas-kelas model yang kaitannya dengan interaksi ke database.
- ✓ **start/**, folder ini merupakan folder yang menyimpan file untuk prosedur aplikasi anda.
- ✓ **storage/**, berfungsi untuk meyimpan file-file yang dibuat oleh Laravel ke harddisk. Misalkan untuk membackup database (file .sql) ke direktori *storage/backups*. Disamping itu, folder ini juga untu menyimpan log apache, sessions dan lainnya.
- ✓ **tests/**, merupakan folder untuk meyimpan semua unit dan tests untuk keperluan pengembangan aplikasi. Secara default laravel akan mencari test dalam folder ini.
- ✓ **views/**, folder yang digunakan untuk meyimpan file-file PHP untuk keperluan tampilan (*Frontend*) aplikasi.
- ✓ **filters.php**, merupakan file yang berisi daftar filer untuk routes aplikasi. Secara default Laravel sudah menyediakan beberapa filder untuk keperluan standar seperti autentifikasi dan proteksi CSRF.
- ✓ **routes.php**, file ini berisi semua route untuk aplikasi khususnya lalu lintas *request* ke aplikasi.

- **bootstrap/**

folder ini berisi tentang file-file prosedur untuk framework laravel. Dalam folder ini terdapat beberapa file yang hanya boleh diedit oleh pengguna laravel yang sudah berpengalaman. Adapaun file-file tersebut adalah sebagai berikut :

- ✓ **autoload.php**, file ini berisi sebagian besar prosedur framework
- ✓ **paths.php**, file ini berisi array dari jalur sistem file umum yang digunakan oleh framework
- ✓ **start.php**, file ini berisi tentang prosedur bagaimana framework laravel berjalan.

- **vendor/**

folder ini berisi tentang semua paket composer yang digunakan diaplikasi kita, file framework laravel juga terdapat dalam folder ini.

- **public/**
folder ini berisi assets yang kita gunakan untuk menyimpan file-file CSS, Javascript, Image ataupun file-file yang diperlukan (biasanya kaitannya erat dengan View – untu membangun frontend).
- **.gitattributes**
File ini merupakan file konfigurasi standar sistem versi kontrol yang sangat populer saat ini yaitu Git.
- **.gitignore**
File ini berisi beberapa informasi folder mana saja yang akan diabaikan oleh Git.
- **artisan**
file ini merupakan file yang berfungsi untuk mengeksekusi atau menjalankan perintah-perintah artisan CLI untuk laravel .
- **composer.json** dan **composer.lock**
kedua file ini berisi informasi tentang paket-paket composer yang akan kita gunakan untuk keperluan aplikasi.
- **phpunit.xml**
file ini berisi konfigurasi default Unit Testing PHP Laravel. File ini juga menangani pemuatan dependensi composer dan mengeksekusi semua test yang ada pada direktori *app/tests*.
- **server.php**
file ini berisi tentang prosedur untuk menlakukan Laravel dengna web server internal yang diperkenalkan pada PHP versi 5.4.

3. Mengetahui Dasar Routing

3.1 Routing Dasar

Hal pertama yang akan kita pelajari yaitu tentang laravel pada buku ini adalah tentang **routing**, apa itu Routing ?

Sebelum beranjak ke materi yang detail tentang routing akan saya coba beri analogi tentang routing ini. Bayangkan jika anda sedang akan login ke facebook, hal pertama yang harus anda lakukan adalah menuliskan url facebook di web browser kemudian enter dan hasilnya akan muncul homepage login facebook. Jika anda mencari sesuatu di google, Anda menulis kata yang dicari di beranda Google kemudian google akan menampilkan hasil dari yang anda cari.

Nah dari analogi diatas, routing digunakan untuk meng-handle **request** yang kita berikan ke aplikasi web. Bisa jadi routing adalah jembatan yang menghubungkan kita dengan respon yang akan diberikan oleh web aplikasi. Jadi setiap ada permintaan (*request*) terhadap alamat tertentu, maka akan alamat akan dieksekusi terlebih dahulu dalam routing sebelum akhirnya akan menampilkan hasil (*response*).

Jika masih bingung tentang routing, akan kita coba langsung di dalam laravel ini.

Buka folder laravel yang telah di instal dengan text editor Anda, kemudian buka route di folder `app/routes.php` . Berikut adalah isian dari `app/routes.php` .

`app/routes.php`

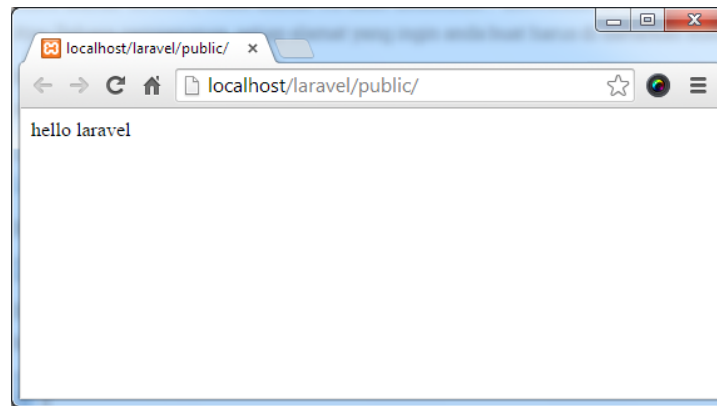
```
Route::get('/', function()
{
    return View::make('hello');
});
```

Ganti respon route diatas menjadi

`app/routes.php`

```
Route::get('/', function()
{
    return 'hello laravel';
});
```

Kemudian buka web browser dan ketikkan alamat `http://localhost/laravel/public` , kemudian hasilnya seperti gambar dibawah ini.

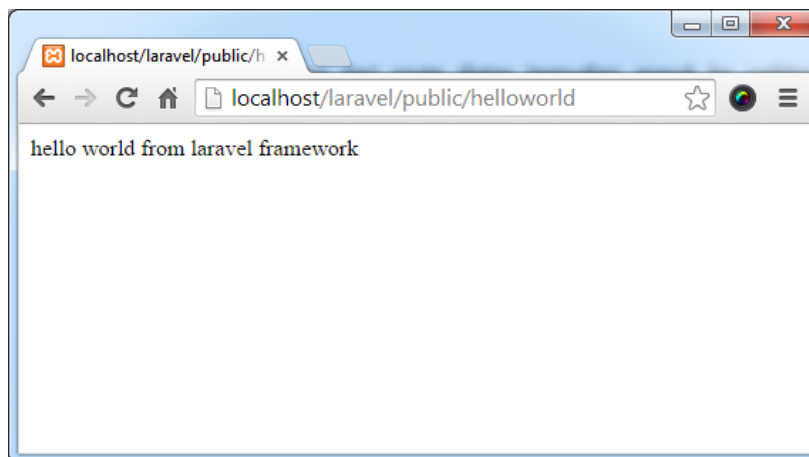


Gambar diatas adalah hasil (*response*) dari route yang telah kita manipulasi tadi. Kemudian kita akan coba membuat route baru dengan mengetikan sintak sebagai berikut.

app/routes.php

```
Route::get('helloworld', function()
{
    return 'hello world from laravel framework';
});
```

Untuk melihat hasilnya dari route diatas kemudian masuk ke webbrowser dan ketik url `http://localhost/laravel/public/helloworld` dan hasilnya adalah sebagai berikut:



Routes selalu dideklarasikan menggunakan kelas `Routes` dan salah satu method yang dipakai untuk *request* sebuah halaman webpage yaitu `GET` menggunakan `HTTP`. `GET request` ini dikirim setiap waktu ketika kita mengetikan sebuah alamat web di webbrowser.

Disamping method `GET`, ada juga method `POST` yang digunakan untuk membuat sebuah permintaan (*request*) dan menyediakan sebuah data yang relatif kecil. Normalnya method ini digunakan sebagai sebuah hasil submit dari form dimana data akan dikirimkan ke database tanpa ditampilkan ke URL.

Ada banyak method yang disediakan oleh kelas `routes` khususnya untuk `restful`, diantaranya.

```
Route::get();  
Route::post();  
Route::put();  
Route::delete();  
Route::any();
```

Kita akan mempelajari method route tersebut di depan khususnya dengan yang berkaitan dengan RESTful routing pada saat proses CRUD (*Create, Read, Update dan Delete*).

3.2 Route Berparameter

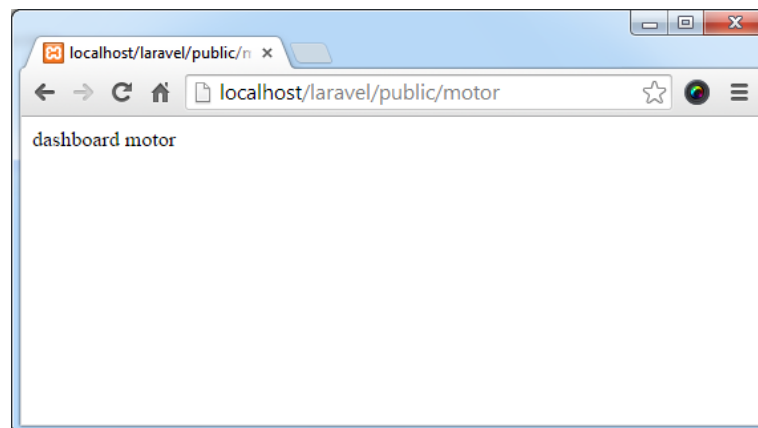
Route berparameter ini dapat digunakan untuk menempatkan sebuah nilai ke route atau URL yang digunakan untuk berbagai keperluan yang dibutuhkan nantinya.

Langsung kita coba, buka file `routes.php` kemudian ketikkan Route baru berikut ini.

app/routes.php

```
Route::get('motor', function()  
{  
    return 'dashboard motor';  
});
```

Route diatas akan menghasilkan hasil sebagai berikut :

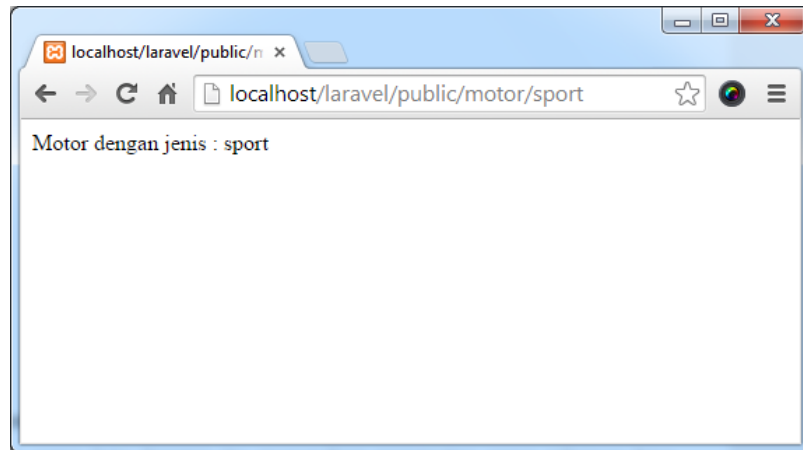


Masih di routes yang sama, kemudian kita akan membuat satu route yang mempunyai parameter yang berfungsi untuk mengirimkan sebuah nilai untuk ditampilkan ke web browser.

app/routes.php

```
Route::get('motor/{jenis}', function($jenis) {  
    return 'Motor dengan jenis : '.$jenis;  
});
```

Kemudian kita ketik URL di browser dan berikan nilai “Sport” untuk route yang berparameter , `http://localhost/laravel/public/motor/sport` dan hasilnya adalah sebagai berikut.



Kamu dapat mencoba dengan berbagai nilai untuk diberikan ke route diatas seperti :

```
http://localhost/laravel/public/motor/bebek  
http://localhost/laravel/public/motor/MotorGede  
http://localhost/laravel/public/motor/RodaTiga
```

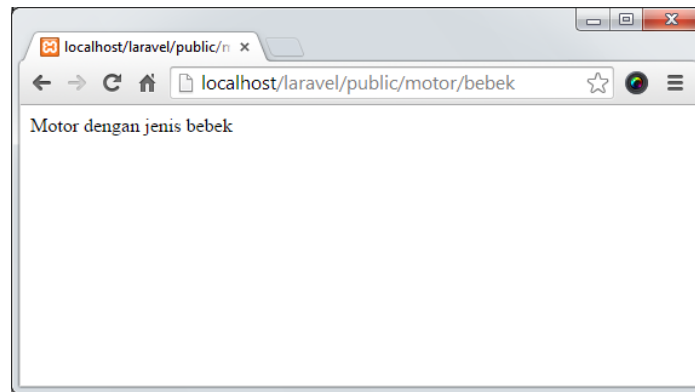
Dalam contoh diatas, kita mengirimkan sebuah nilai yaitu “Sport” pada route berparameter `motor/{jenis}` dan dalam function route tersebut kita deklarasikan variabel `$jenis` untuk ditampilkan pada saat *response*.

Sebuah parameter juga dapat dijadikan sebuah pilihan jika parameter tersebut tidak diisi (null) atau diberi nilai *default* maka dapat ditambahkan sebuah tanya tanya (?)

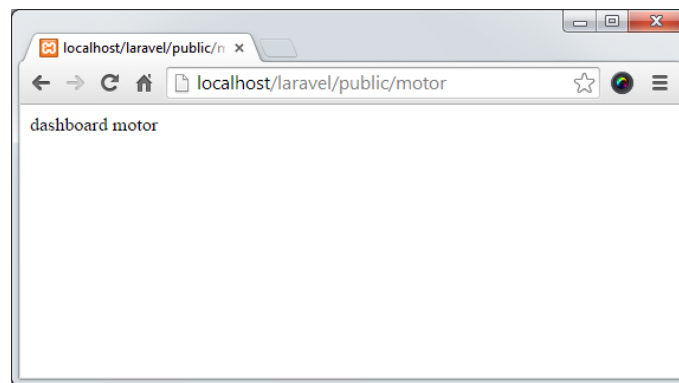
app/routes.php

```
Route::get('motor/{jenis?}', function($jenis=null) {  
    if($jenis == null) return "Motor Dashboard Page";  
    return "Motor dengan jenis ".$jenis;  
});
```

Jika route diatas ada fungsi logika nya yaitu jika nilai `$jenis = null` (kita tidak memberikan nilai di URL) maka akan mengembalikan "Motor Dashboard Page";.Tapi jika variabel `$jenis` kita beri nilai maka akan mengembalikan "Motor dengan jenis "<nilai_variabel>;. Kita dapat lihat pada contoh sebagai berikut.



Tapi kalau kita biarkan atau kita tidak mengisi nilai “bebek” untuk URL diatas, maka hasilnya adalah sebagai berikut :



Disamping nilai default (null), kita juga dapat memberikan nilai sesuai dengan kebutuhan. Misalkan kita beri contoh default untuk variabel `$jenis = Sport`, maka route nya seperti ini.

app/routes.php

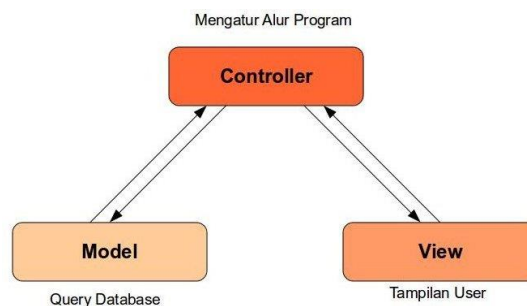
```
Route::get('motor/{jenis?}', function($jenis=Sport) {  
    return "Motor dengan jenis ".$jenis;  
});
```

Route diatas apabila kita eksekusi dengan tidak memberikan nilai di URL nya maka akan mengembalikan *response* dengan nilai "Motor dengan jenis Sport ". Tapi kalau kita memberikan nilai di URLnya misalkan “Bebek”, maka hasil responnya adalah sebagai berikut "Motor dengan jenis Bebek".

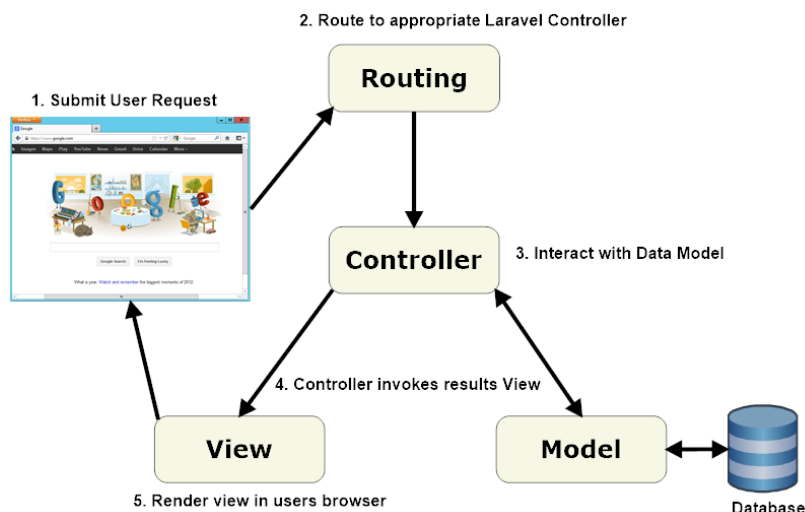
4. Mengenal MVC (Model-View-Controller)

MVC atau kepanjangan dari Model-View-Controller adalah sebuah metode yang digunakan dalam pengembangan suatu aplikasi yang memisahkan data (model) dari tampilan / *frontend* (View) dan *logic* dari aplikasi itu sendiri (Controller). MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna dan kontrol dalam sebuah aplikasi.

Model digunakan untuk proses query atau manipulasi data ke/dari database. Sedangkan **View** kaitannya erat dengan antarmuka / *frontend* tampilan sebuah web seperti HTML, CSS dan JS dan data yang bersifat *client*. **Controller** adalah logika dari sebuah web. Menjembatani komunikasi antara **Model** dan **View**. Kalau digambarkan alur proses **MVC** adalah sebagai berikut :

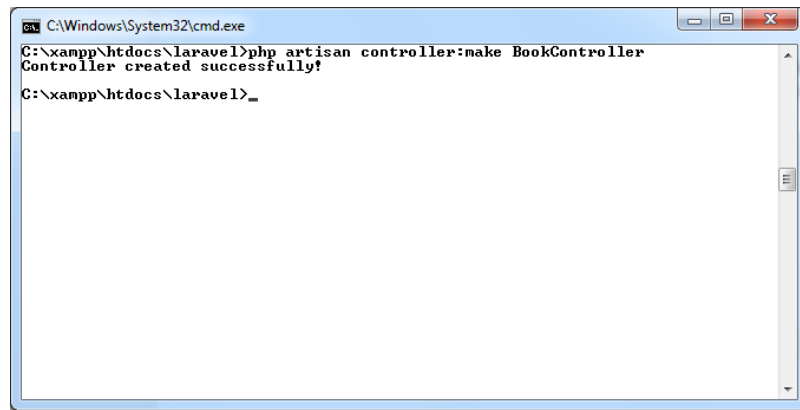


Adapun gambar simulasi proses MVC pada Laravel diperlihatkan pada gambar dibawah ini.



Untuk lebih jelasnya kita langsung praktikan proses MVC pada laravel. Pertama kita membuat sebuah controller dengan nama **BookController**. Disini saya menggunakan composer untuk membuat controller. Dengan sintak sebagai berikut :

```
php artisan controller:make BookController
```



Setelah itu **BookController** isi sebagai berikut :

app/controllers/BookController.php

```
<?php

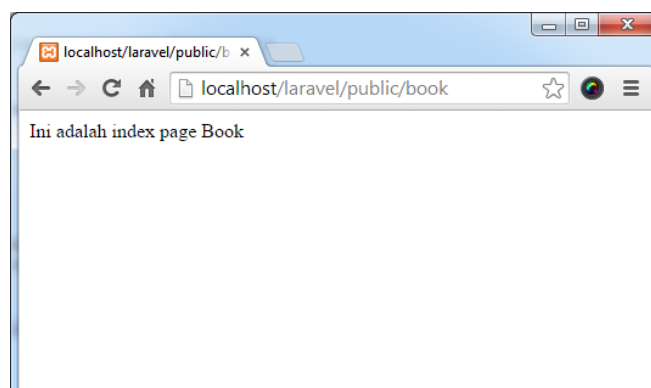
class BookController extends \BaseController {
    public function index() {
        return "Ini adalah index page Book";
    }
}
```

Setelah membuat controller, kemudian kita buka file `routes.php`. Buat sebuah route yang merujuk pada controller yang telah dibuat diatas.

app/routes.php

```
Route::get('book', 'BookController@index');
```

Penjelasan route diatas adalah Route menggunakan method `get` untuk menampilkan *response* dari method `index` dari controller `BookController`. Untuk melihat output dari proses diatas ketikkan URL sebagai berikut `http://localhost/laravel/public/book` . Adapun hasilnya seperti gambar dibawah ini.



Kita akan coba bagaimana mengirimkan sebuah parameter / nilai dari route ke controller. Ganti route book kemudian tambahkan parameter “judul”.

app/routes.php

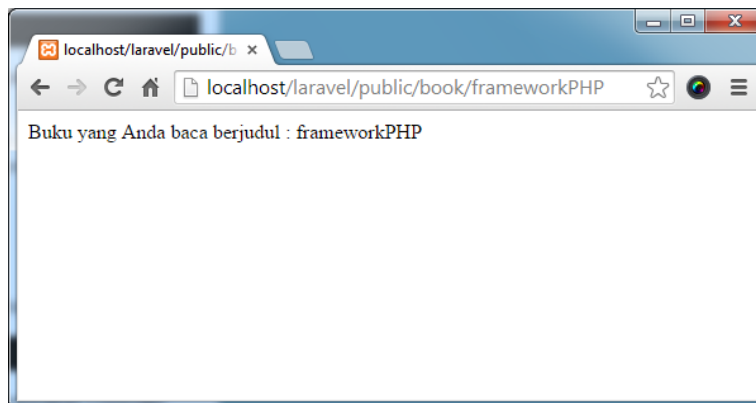
```
Route::get('book/{judul}', 'BookController@viewJudul');
```

Lalu kita buat satu buah method **viewJudul** di **BookController**

app/controllers/BookController.php

```
public function viewJudul($judul) {  
    return "Buku yang Anda baca berjudul : ".$judul;  
}
```

Kemudian cek kode diatas dengan mengetikan URL sebagai berikut pada web browser `http://localhost/laravel/public/book/frameworkPHP`



Setelah itu, kita akan coba mengintegrasikan **Controller** dengan **View**. Pada folder **app/view** kemudian buat sebuah file PHP dengan nama **Book.php** dan isikan kode sebagai berikut.

app/view/Book.php

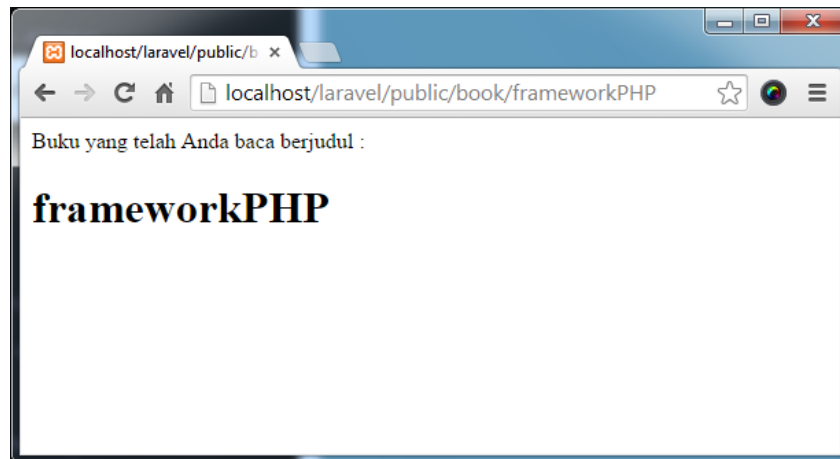
```
<h1>Page View</h1>  
Buku yang telah Anda baca berjudul : <h1><?php echo $judul ?></h1>
```

Buka controller **BookController** kemudian edit method **viewJudul** dan isikan kode sebagai berikut.

app/controllers/BookController.php

```
public function viewJudul($judul) {  
    $data=array(  
        'judul'=>$judul  
    );  
    return View::make('Book',$data);  
}
```

Method **viewJudul** diatas akan mengeksekusi view **Book.php** pada folder **app/view**. Dan kalau kita jalankan di web browser dengan mengetikan URL sebagai berikut <http://localhost/laravel/public/book/frameworkPHP> hasilnya adalah sebagai berikut.



untuk bagian **Model** nanti saya akan bahas beserta manajemen database (*migration & seeder*)

5. Mengenal Blade Template Engine Laravel

Template engine adalah sebuah method untuk mempersingkat penulisan kode yang lebih panjang contoh nya yaitu smartt, twig dan doo. Sedangkan **blade** itu sendiri adalah template engine bawaan laravel. Blade menawarkan penulisan kode/sintax yang mudah dan singkat untuk dipakai dalam menghasilkan kode HTML.

Pada bagian **View** inilah fungsi **Blade** sangat dibutuhkan. **View** seperti yang sudah kita tahu berfungsi menampilkan sebuah halaman web, namun bukan berarti dalam view tersebut tidak bisa melakukan proses *logic*. Disinilah peran blade yang dibutuhkan untuk membantu menuliskan *logic* agar menjadi lebih *simple*. Disamping itu, blade juga berfungsi untuk memisahkan *layout* suatu web dengan *layout* tertentu dan blade sendiri mendukung *inheritance* (OOP). Semua file blade harus menggunakan ekstensi **.blade**. Contoh jika kita membuat sebuah file Book.php maka untuk bisa menggunakan fitur blade, maka harus diberi nama menjadi **Book.blade.php** . Berikut adalah perbedaan mendasar antara sintak PHP dan blade.

PHP Syntax	Blade Syntax
<?php echo \$var; ?>	{{ \$var }}
<?php echo htmlentities(\$var); ?>	{{{ \$var }}}}
<?php if(\$cond) : ?> .. <?php endif; ?>	@if(\$cond) .. @endif

Blade juga mendukung penuh proses looping dan kondisi-logika PHP seperti @for, @foreach, @while, @if dan @elseif. Supaya tidak bingung dalam menuliskan sintak berikut akan saya contohkan menulis sintak php biasa dengan sintak blade.

echo variabel

```
/*script php*/
<?php echo "Halo ini cara lama" ?>

/* script blade template*/
{{ "Halo ini cara blade template" }}
```

echo variabel dengan escape html

```
/*script php*/
Halo ini cara lama <?php echo htmlspecialchars($name, ENT_QUOTES, 'UTF-8')
?>

/*cara baru dengan menggunakan blade template*/
Halo ini cara blade template {{{ $name }}}}
```

echo variabel dengan nilai default

```
/*script php*/
Halo ini cara lama <?php echo isset ($name)?($name): 'guest' ?>

/*cara baru dengan menggunakan blade template*/
Halo ini cara blade template {{ $name or 'guest' }}
```

kondisi

```
/*script php*/
<?php
if (status==0) {
    echo "Proses Gagal";
} elseif (status==1) {
    echo "Proses Berhasil";
} else {
    echo "Tidak diketahui";
}
?>

/*cara baru dengan menggunakan blade template*/
@if (status==0)
    {{ "Proses Gagal" }}
@elseif (status==1)
    {{ "Proses Berhasil" }}
@else
    {{ "Tidak diketahui" }}
@endif

/*kebalikan dari if, jika kondisi tidak memenuhi syarat yang ada*/
@unless($isLogin)
    {{ "Anda tidak berhak mengakses halaman ini" }}
@endunless
```

Looping atau Iterasi

```
/*script php*/
/* for statement*/
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
?>

/* while statement */
<?php while ($i <= 10) :
    echo $i++;
endwhile;
?>

/* foreach statement */
<?php
foreach($arr as $value)
{
    $value=$value + 1;
    echo $value;
}
?>

/*cara baru dengan menggunakan blade template*/
/* for statement */
@for ($i = 1; $i <= 10; $i++)
    {{ $i }}
@endfor

/* while statement */
@while ($i <= 10)
```

```

    {{ $i ++ }}
@endwhile

/* foreach statement */
@foreach($arr as $value)
    {{ $value = $value + 1; }}
    {{ $value }}
@endforeach

```

include sub-view

```

/*script php*/
include 'folder/subview';

/*cara baru dengan menggunakan blade template*/
@include('folder.subview')

```

Membuat Form HTML

```

/*script php*/
<form role="form" method="post" action="name_url">
    <input type="text" name="kode" placeholder="kode"/>
    <input type="submit" name="submit" class="btn btn-primary"/>
</form>

/*cara baru dengan menggunakan blade template*/
{{ form::open(array('url'=>'name_url','method' => 'post')) }}
{{ Form::text('kode', Input::old('kode'), array('class' => 'form-control','placeholder'=>'Kode')) }}
{{ Form::submit('Proses', array('class' => 'btn btn-primary')) }}
{{ form::close() }}

```

memanggil file CSS dan JSS

```

/*cara baru dengan menggunakan blade template*/
/*memanggil script file *.js*/
{{HTML::script('name_file.js')}}

/*memanggil script file *.css*/
{{HTML::style('name_file.css')}}

```

Diatas adalah perbedaan-perbedaan mendasar antara PHP sintak dengan blade sintak. Disamping itu, blade juga mendukung *reusable-coding* atau penggunaan kembali. Hal ini sesuai dengan paradigma pemrograman berorientasi objek yaitu *inheritance* dimana kita tidak perlu membuat kode berulang-ulang pada halaman yang berbeda, adapun fitur tersebut yaitu *template inheritance* dan *section*. Berikut akan saya beri contoh tentang penggunaan blade.

Buat file di folder **app/view** dengan nama **default.blade.php** kemudian isikan kode berikut.

app/view/default.blade.php

```
<html lang="en">
  <head>
    <title>Blade Template</title>
    {{
HTML::style('https://maxcdn.bootstrapcdn.com/bootstrap/3.3.0/css/bootstr
ap.min.css') }}
  </head>
  <body>
    <div class="container">
      <div class="jumbotron">
        <h1>Blade Template - Laravel</h1>
        <p>@yield('content')</p>
        <p><a href="#" class="btn btn-primary btn-lg" >Learn more
&raquo;</a></p>
      </div>
    </div>
  </body>
</html>
```

Masih di folder **app/view** kemudian buat file **index.blade.php** dan masukan sintak berikut

app/view/index.blade.php

```
@extends('default')
@section('content')
    {{ "Gettig Started with Blade Template " }}
@stop
```

Masuk ke folder **kontroller** kemudian buka **HomeController** tambahkan method **index**.

app/controllers/HomeController.php

```
public function index() {
    return View::make('index');
}
```

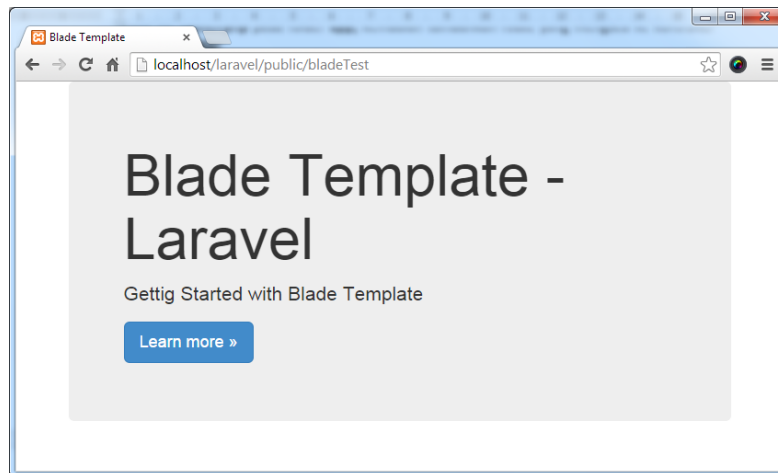
Buka file **routes.php** pada folder **app**, kemudian tambahkan route yang mengacu ke **kontroller HomeController**

app/routes.php

```
Route::get('bladeTest', 'HomeController@index');
```

Kemudian buka browser dan ketikan URL nya

<http://localhost/laravel/public/bladeTest> jika berhasil, maka hasilnya akan sebagai berikut.



6. Form dan HTML

Bagi yang sudah terbiasa membuat program website khususnya pasti sudah mengenal apa itu form atau lebih dikenal tag `<form>`. Form berguna untuk menangani berbagai inputan dari *user* misalkan untuk mengisi identitas orang, data buku dan lain-lain.

Di laravel, ada penangan khusus jika anda ingin menggunakan Form dengan sintak yang lebih sederhana dan tentunya dengan menggunakan fitur **blade** sendiri. Meskipun begitu, laravel juga mengizinkan developer untuk menuliskan sintak murni HTML Form untuk mengelolanya.

Pada bab ini, kita akan mempelajari bagaimana membuat form dan integrasi dengan route dan kontroller.

6.1 Opening Form

Sebelum kita menuliskan tag form komponen, terlebih dahulu kita membuat atau membuka sebuah tag form. Adapun di laravel untuk membuka sebuah tag form dengan sintak sebagai berikut :

```
{{ Form::open() }}  
//  
{{ Form::close() }}
```

Sintak diatas apabila kita jalankan di browser, maka akan menghasilkan kode HTML sebagai berikut :

```
<form method="POST" action="#" accept-charset="UTF-8">  
</form>
```

Default method untuk form adalah POST dengan link/*action* default (#) dan menggunakan font-default yaitu **UTF-8** .

Mengarahkan Form ke URL

Bila kita ingin mengarahkan form ke URL tertentu, kita bisa menggunakan sintak ini :

```
{{ Form::open(array('url'=>'http://www.cobalaravel.com/form')) }}
```

Mengarahkan Form ke Route

Disamping ke URL, kita juga dapat mengarahkan form ke Route yang sudah terdaftar. Adapapun sintaknya sebagai berikut :

```
{{ Form::open(array('route'=>'book/save')) }}
```

Mengarahkan Form ke Kontroller

Form juga dapat diarahkan langsung ke kontroller. Berikut adalah sintaknya :

```
{{ Form::open(array('action'=>'BookController@save')) }}
```

Menggunakan Method yang Berbeda

Secara default, jika kita tidak mendefinisikan method, maka laravel akan menempatkan method default yaitu POST. Jika kita ingin mendefinisikan method yang berbeda berikut adalah sintaknya :

```
{{ Form::open(array('method'=>'GET')) }}  
{{ Form::open(array('method'=>'PUT')) }}  
{{ Form::open(array('method'=>'PATCH')) }}  
{{ Form::open(array('method'=>'DELETE')) }}
```

Menggunakan Class CSS untuk Form

Jika memasukkan atribut CSS misalkan `class` atau `id` ke dalam form, sintaknya sebagai berikut:

```
{{ Form::open(array('class'=>'container')) }}  
{{ Form::open(array('id'=>'container')) }}
```

Setelah kita mengetahui dasar sintak form, misalkan kita membuat sebuah form dengan method POST dan diarahkan ke route `book/save` dengan class `form-control`, adalah sebagai berikut.

```
{{ Form::open(array('method'=>'POST','route'=>'book/save','class'=>'form-control')) }}
```

6.2 Form Binding / Model

Bagian ini digunakan biasanya pada saat kita ingin memperbarui / *update* data. Fungsi ini mem-*binding* data dari model. Adapun sintaknya sebagai berikut:

```
{{ Form::model($data,array('route'=>'update',$item->id)) }}
```

Penjelasan :

- `$data` , adalah variabel yang dikirim dari Kontroller atau Route. Biasanya memiliki seluruh record data yang berasal dari table database untuk ditampilkan di form (Proses Query Database).
- Untuk parameter kedua `array()` , semua inputan dalam form akan diberikan ke route `update` sambil mengirim id dari record yang ingin di update.

6.3 Form Komponen

Setelah kita mengenal properti dari Form Laravel, selanjutnya kita akan mempelajari komponen dari Form itu sendiri seperti: text, password, textarea dan lain-lain.

Label

Untuk menggenerate sebuah label, kita dapat menggunakan sintak sebagai berikut :

```
{{ Form::label('email','Email')}}}
```

Text

Untuk membuat sebuah inputan teks, sintaknya sebagai berikut :

```
{{ Form::text('email','Email')}}}
```

Text Area

untuk membuat textarea, gunakan sintak nya sebagai berikut:

```
{{ Form::textarea('alamat','Alamat')}}}
```

Password

Untuk membuat inputan password dan *fields* yang tersembunyi adalah sebagai berikut:

```
{{ Form::password('password','Password')}}}
```

Checkbox

Checkbox mengijinkan form untuk mendapatkan nilai *boolean*.

```
{{ Form::checkbox('isRead','1',true)}}}
```

Parameter pertama adalah untuk mengisi atribut *name* . Parameter kedua adalah nilai dari *field* dan parameter yang ketiga adalah nilai yang menentukan apakah checkbox akan di check atau tidak di check. Berikut adalah bentuk dari tag HTML diatas.

```
<input checked="checked"
  name="isRead"
  type="checkbox"
  value="1"
  id="isRead">
```

Radio Button

Radio button digunakan untuk memilih satu pilihan dari beberapa pilihan yang disediakan. Berikut adalah sintaknya yang digunakan :

```
{{ Form::radio('color','red',true) }} Red
{{ Form::radio('color','black') }} Black
{{ Form::radio('color','blue') }} Blue
```

Parameter pertama yaitu untuk menentukan *class* dan *id* dari radiobutton yaitu `color`. Parameter kedua untuk menentukan nilai dari tiap-tiap radio button sedangkan parameter ketiga adalah untuk menentukan *default* dari pilihan mana yang sudah di check. Apabila kita konversi ke bentuk biasa kode diatas adalah sebagai berikut :

```
<input checked="checked"
      name="color"
      type="radio"
      value="red"
      id="color"> Red
<input name="color"
      type="radio"
      value="black"
      id="color"> Black
<input name="color"
      type="radio"
      value="blue"
      id="color"> Blue
```

SelectBox

Selectbox atau dropdown digunakna untuk menyediakan sebuah pilihan dari beberapa pilihan yang disediakan. Berikut adalah sintaknya

```
{{ Form::select('color',array(
    'red' => 'Red',
    'black' => 'Black',
    'blue' => 'Blue'
),'red') }}
```

Parameter pertama untuk mengisi nilai dari attribut *class* dan *id* yaitu *color*. Parameter kedua yaitu tipe *array* untuk mengisi nilai di selectbox tersebut. Sedangkan parameter ketiga yaitu untuk mengisi nilai *default* dari selectbox yaitu *red*. Kode diatas jika dikonversikan ke bentuk HTML yaitu :

```
<select id="color" name="color">
    <option value="red" selected="selected">Red</option>
    <option value="black">Black</option>
    <option value="blue">Blue</option>
</select>
```

Email Field

Tipe email ini sama seperti input text. Perbedaannya adalah bahwa `Form::email()` adalah elemen baru dari HTML5 yang akan memvalidasi bahwa *user* harus memastikan bahwa nilai yang diinputkan harus email yang *valid*.

```
{{ Form::email('email','me@email.com') ;
```

Parameter pertama digunakan untuk mengisi atribut *name* dan parameter kedua adalah nilai dari email tersebut.

File Upload File

Untuk keperluan upload file, laravel menyediakan cara untuk upload file.

```
{{ Form::file('avatar') ;
```

Tombol Submit

Adapun tombol submit di Form laravel adalah sebagai berikut :

```
{{ Form::submit('Save') ;
```

7. Schema Builder Dasar

Mulai dari bab ini, kita akan sering berhubungan dengan database. Saya asumsikan anda telah mengetahui tentang skema database table dan komom serta relasi tabel, data tipe dan lain-lainya. Dalam bab ini juga kita akan mempelajari tentang kelas `Schema` yang akan kita gunakan untuk membuat struktur dari tabel yang akan dibuat di database.

Sebelumnya kita masuk ke materi, kita buka dulu file file database.php yang ada dalam direktori **app/config/database.php** . Disini kita menggunakan *engine database* nya yaitu MySQL. Dalam file tersebut pada pilihan **connections** atur konfigurasi database mysql menjadi .

app/config/database.php

```
'mysql' => array(
    'driver' => 'mysql',
    'host' => 'localhost',
    'database' => 'db_laravel',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix' => '',
),
```

Kemudian masuk ke database mysql dan buat database “db_laravel”.

7.1 Membuat Tabel

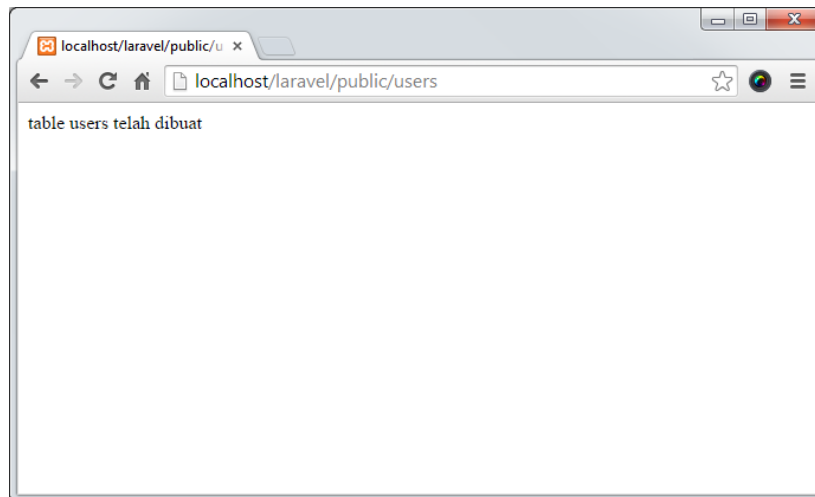
Buka file routes.php dan buat route baru untuk membuat sebuah tabel.

app/routes.php

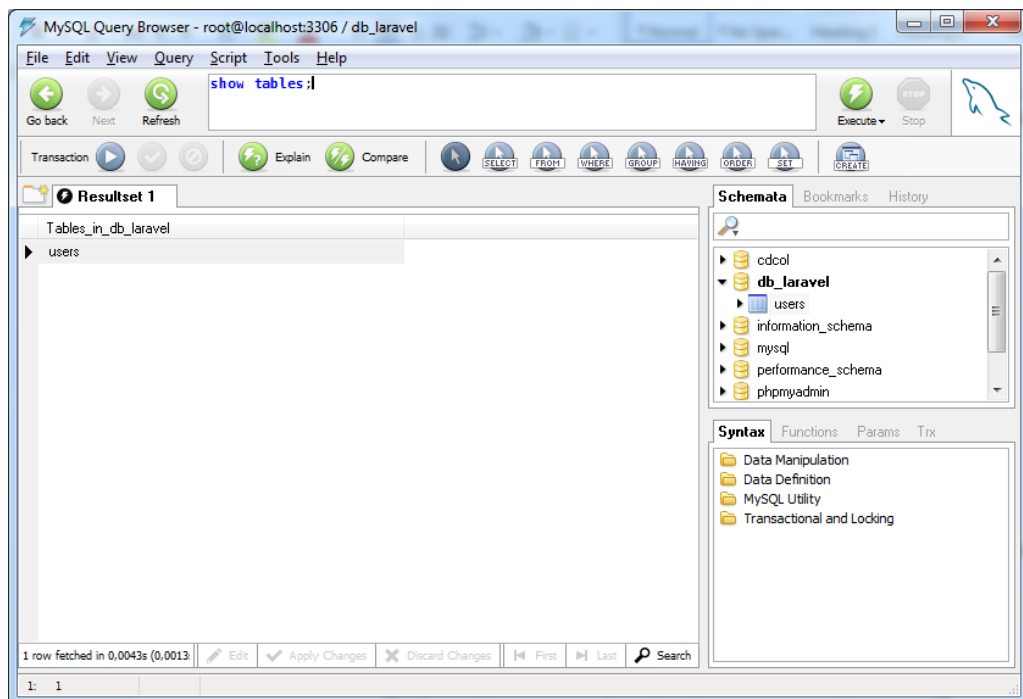
```
Route::get('users', function() {
    Schema::create('users', function($table) {
        $table->increments('id');
        $table->string('useranme', 32);
        $table->string('email', 320);
        $table->string('password', 60);
        $table->timestamps();
    });
    return "table users telah dibuat";
});
```

`Schema::create` mempunyai dua parameter. Parameter pertama, yaitu untuk mendefinisikan nama tabel yang akan dibuat sedangkan parameter kedua yaitu untuk mendefinisikan struktur tabel dari tabel user.

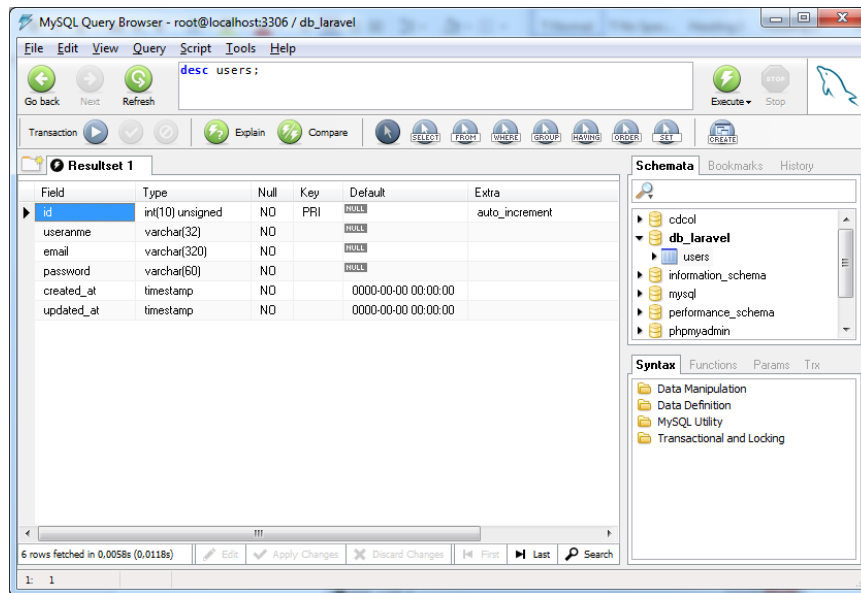
Untuk mengeksekusi kode diatas kemudian buka browser dan ketikan URL dibawah ini <http://localhost/laravel/public/users> .



masuk ke database mysql, pilih database **db_laravel**, untuk mengecek apakah tabel telah dibuat kemudian ketik sintak “show tables”.



Tabel users telah dibuat di mysql, kemudian ketikan sintak “desc users”, untuk mengetahui struktur dari tabel.



7.2 Menambah, Rename dan menghapus kolom

Kadang-kadang di tengah proses *development* suatu aplikasi, adakalanya kita ingin menambahkan, menghapus atau bahkan mengganti nama dari suatu kolom. Berikut adalah sintak untuk manipulasi kolom.

Tambah kolom

```
Schema::table('users', function($table) {
    $table->text('alamat');
});
```

Rename kolom

Untuk mengubah kolom dapat menggunakan perintah **renameColumn**

```
Schema::table('users', function($table) {
    $table->renameColumn('alamat', 'alamatKirim');
});
```

Parameter pertama yaitu kolom yang sudah disimpan di dalam database sedangkan parameter kedua yaitu nama baru yang akan disimpan untuk mengganti nama kolom.

Menghapus Kolom

Untuk menghapus kolom dapat menggunakan perintah sebagai berikut :

```
Schema::table('users', function($table) {
    $table->dropColumn('alamat');
});
```

Untuk menghapus lebih dari satu kolom gunakan sintak berikut

```
Schema::table('users', function($table) {
    $table->dropColumn(array('alamat', 'notelp'));
});
```


7.3 Menambahkan Index dan Foreign Key

Hal ini sangat berguna jika ada tabel yang berelasi dengan tabel lain dengan cara menambahkan foreign key atau menambahkan hal yang unik untuk kolom dari suatu tabel.

Menambahkan index pada kolom

```
$table->string('alamat')->unique();
```

Menambahkan Foreign Key

```
$table->integer('id_user')->unsigned();  
$table->foreign('id_user')->references('id')->on('users');
```

Penjelasan kode diatas adalah membuat kolom `id_user` dimana kolom tersebut berelasi dengan kolom `id` yang ada di tabel `users`.

7.4 Tipe Kolom

Dari tabel user yang telah kita buat diatas, ada tipe data **increments**, **string** dan **timestamps** merupakan tipe data yang nantinya akan dikonversikan ke tipe data yang ada dalam database mysql.

- **increments**

tipe ini akan memberikan nilai integer dengan nilai yang bertambah secara otomatis (*increments*)

```
$table->increments('id');
```

- **bigIncrements**

jika dirasa tipe **increments** tidak cukup untuk kamu. Bisa menggunakan method **bigIncrements()** yang akan membuat tipe data big integer.

```
$table->bigIncrements('id');
```

- **string**

method ini digunakan untuk menghasilkan tipe data **varchar** di mysql.

```
$table->string('username', 32);
```

Parameter pertama untuk memberi nama kolom / *field* sedangkan kolom kedua untuk memberi rentang nilai yang diberikan untuk kolom tersebut.

- **text**

method text ini digunakan untuk menyimpan data teks yang berukuran besar seperti menyimpan artikel, berita ataupun postingan dari blog.

```
$table->text('post');
```

- **integer**

biasanya digunakan untuk tipe data yang berupa bilangan.

```
$table->integer('age',3);
```

Parameter pertama digunakan untuk memberi nama kolom sedangkan parameter kedua digunakan untuk memberi rentang nilai kolom.

- **float**

float digunakan untuk menyimpan bilangan pecahan atau desimal.

```
$table->float('size');
```

- **boolean**

tipe ini hanya mempunyai nilai *true* atau *false*.

```
$table->boolean('isSmart');
```

- **date**

sesuai dengan namanya, tipe ini digunakan untuk menyimpan tanggal / *date*

```
$table->date('departure');
```

- **timestamps**

method ini digunakan untuk menyimpan data tanggal dan waktu dalam format TIMESTAMP. Pada tabel user diatas, kolom timestamps akan menghasilkan dua kolom di tabel users mysql yaitu `created_at` dan `updated_at`.

```
$table->timestamps();
```

8. Mengetahui Migrations

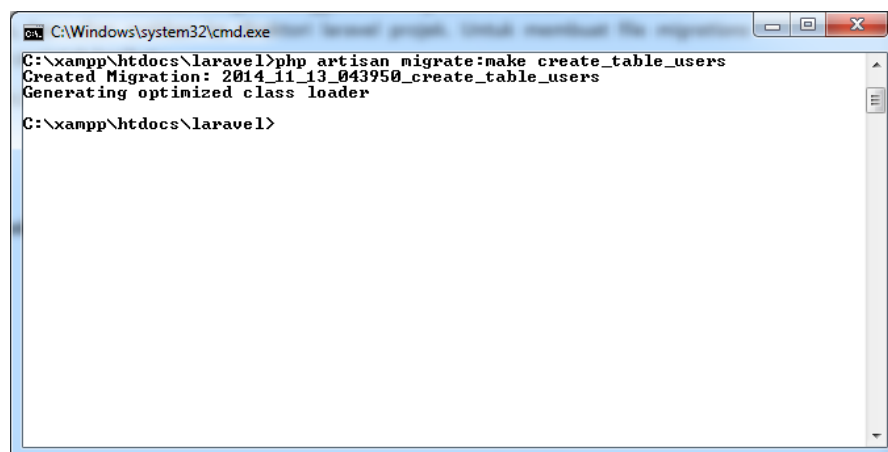
Kita membuat struktur tabel, relasi dan lain-lainnya biasanya langsung dari mysql langsung atau menulis sintak SQL dan mendeskripsikan tabel beserta kolom apa saja yang dibutuhkan, tapi apa yang akan terjadi jika kita secara tidak sengaja menghapus database tersebut ? apa yang akan terjadi jika kamu belajar sebagai team ? mungkin kamu akan memberikan SQL dump ke masing-masing anggota untuk menjaga database agar selaras.

Dari permasalahan dasar seperti itulah fungsi *migrations* sangat dibutuhkan. Dengan menggunakan fitur *migrations* ini, kita dapat membuat, memodifikasi dan menghapus suatu tabel atau relasi antar tabel dengan menggunakan kode program dari laravel itu sendiri yaitu *migrations*. Dengan menggunakan *migrations*, kamu dan tim dan menjaga konsistensi struktur database, tabel-tabel beserta kolomnya. Atau jika masih bingung dari penjelasan diatas, kita akan langsung coba mempraktekannya.

8.1 Membuat Migrations

Masih menggunakan konfigurasi database yang ada di bab **Schema Builder**, pertama hapus terlebih dahulu tabel **"users"** yang telah tersimpan di database MySQL karena kita akan membuat tabel user kembali dengan menggunakan *migrations*. Masuk ke Command Prompt windows, kemudian arahkan ke direktori laravel proyek. Untuk membuat file *migrations* gunakan perintah berikut :

```
php artisan migrate:make <nama_file_migrations>
```



Jika hasilnya sama dengan gambar diatas, kemudian kita cek pada direktori **app/database/migrations** disini ada file migrations php dengan nama **2014_11_13_043950_create_table_users.php** . nama awalnya bisa berbeda-beda tapi nama akhirnya pasti sama. Dan apabila kita buka, akan menghasilkan kode php sebagai berikut :

app/database/migrations/2014_11_13_043950_create_table_users.php

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateTableUsers extends Migration {

    /**
     * Run the migrations.
     */
}
```

```

        * @return void
        */
        public function up()
        {
            //
        }

        /**
         * Reverse the migrations.
         *
         * @return void
         */
        public function down()
        {
            //
        }
    }
}

```

Disana terdapat method `up()` dan `down()`. Dimana method `up()` digunakan untuk membuat table dan memanipulasi kolom dari tabel sedangkan method `down()` biasanya digunakan untuk menghapus tabel atau kolom.

Pada method `up()` dan `down()` kemudian kita definisikan tabel users beserta kolom nya .

app/database/migrations/2014_11_13_043950_create_table_users.php

```

public function up()
{
    Schema::create('users', function($table) {
        $table->increments('id');
        $table->string('nama', 128);
        $table->string('email');
        $table->string('password', 60);
        $table->timestamps();
    });
}

public function down() {
    Schema::drop('users');
}

```

8.2 Menjalankan Migrations

Setelah itu masuk ke command prompt yang tadi, kemudian kita akan menjalankan file *migration* diatas dengan sintak sebagai berikut, jika ada konfirmasi install, ketik (y) kemudian Enter :

```
php artisan migrate
```

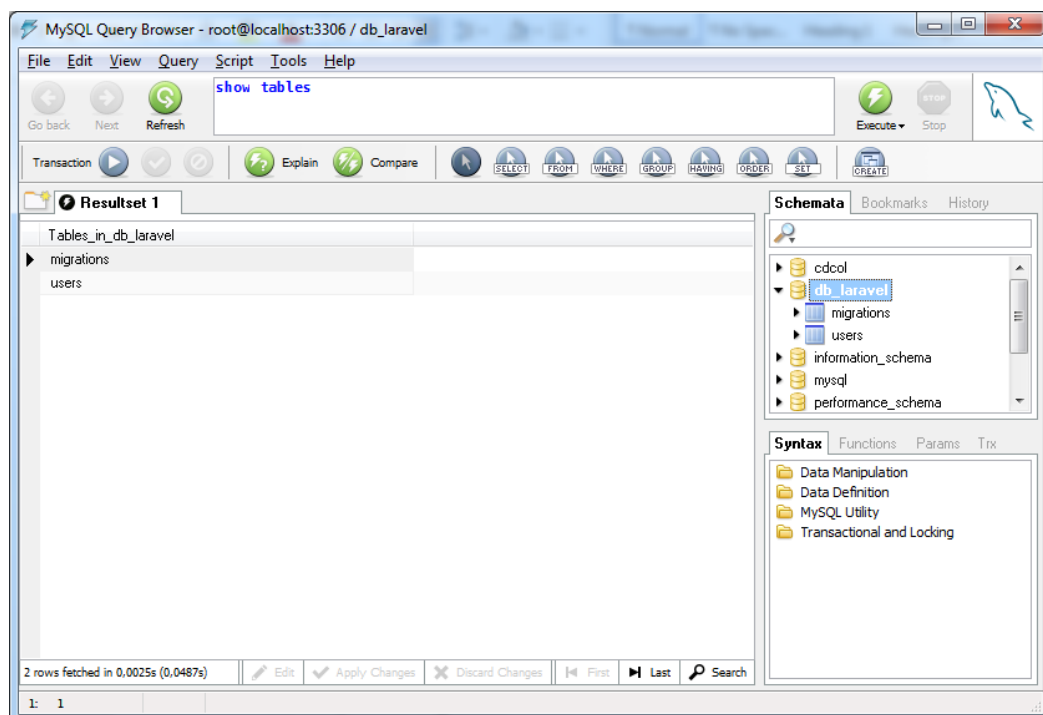
```
C:\Windows\system32\cmd.exe

C:\xampp\htdocs\laravel>php artisan migrate
*****
*      Application In Production!      *
*****

Do you really wish to run this command? y
Migration table created successfully.
Migrated: 2014_11_13_043950_create_table_users

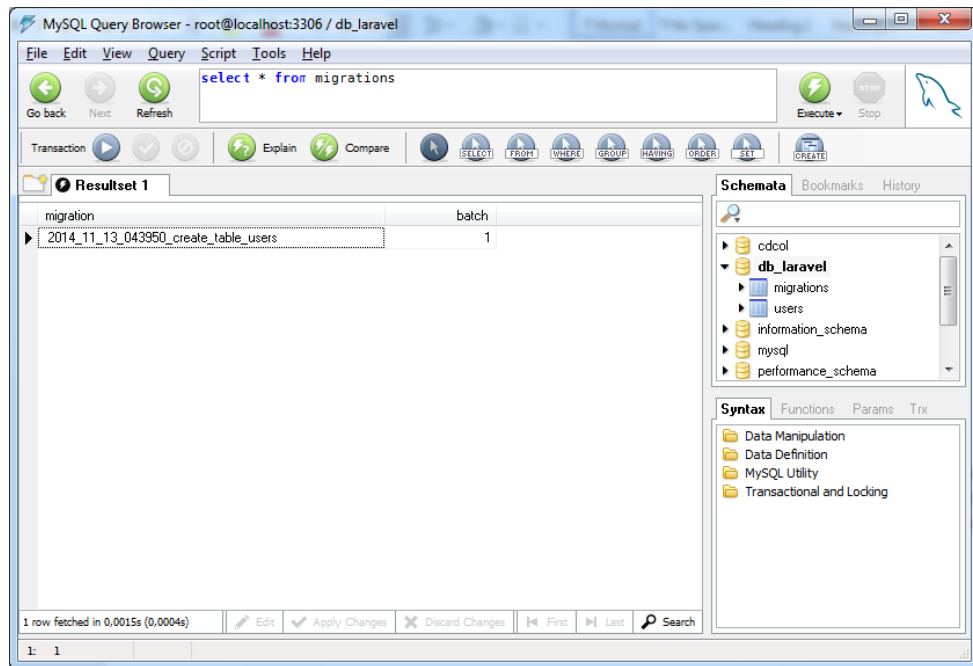
C:\xampp\htdocs\laravel>
```

File tersebut sudah berhasil, kemudian masuk ke database mysql untuk mengeceknya.



Diatas, pada database **db_laravel** terdapat dua tabel yaitu tabel **migrations** dan **users**. Karena kita menggunakan fitur *migrations*. Maka otomatis laravel akan membuat tabel migrations yang isinya tentang tabel-tabel yang sudah dimigrasi ke MySQL. Untuk mengeceknya coba tuliskan sintak berikut di MySQL.

```
select * from migrations
```



Pada perintah `php artisan migrate` diatas, akan mengeksekusi semua file *migrations* yang ada pada direktori `app/database/migration`, tapi jika kita hanya ingin mengeksekusi hanya satu file *migrations*, gunakan perintah dibawah ini.

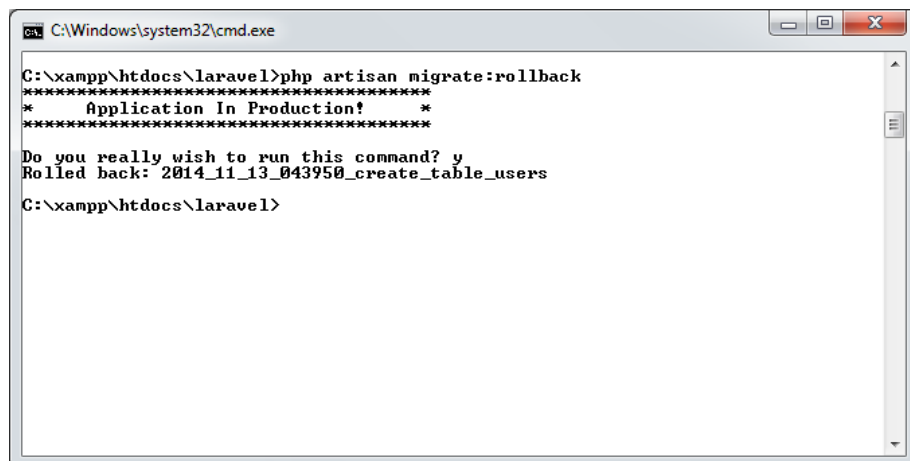
```
php artisan migrate --path=app/database/migrations/<nama_file>
```

8.3 Rolling Migrations

Seperti yang telah kita ketahui, *migrations* digunakan untuk kepentingan struktur tabel atau relasi. Tapi kita asumsikan, kita sedang dalam suatu kondisi yang mengharuskan kita mengatur ulang tabel-tabel yang telah kita buat.

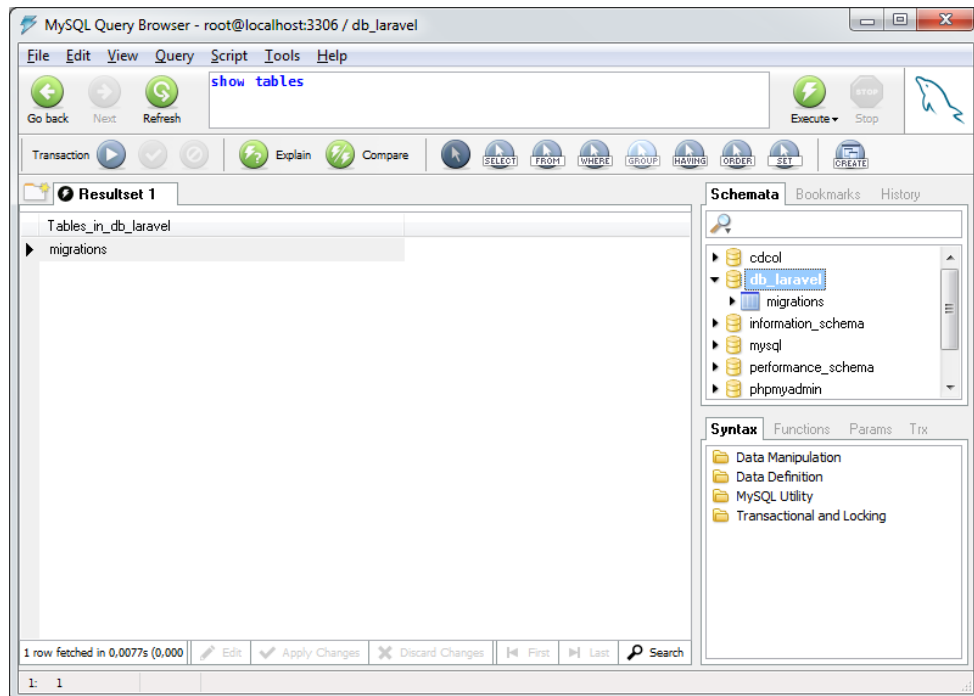
Dari situ, kita membutuhkan untuk me-rollback perubahan dari tim yang sudah dibuat. Maka kita dapat menggunakan perintah *rollback*.

```
php artisan migrate:rollback
```



Jika kita mengecek didatabase db_laravel, maka tabel yang suda dibuat menggunakan migrasi akan dibalikn kembali pada saat kita terakhir kali menggunakan perintah **migrate** . Atau jika kita ingin me-*rollback* semua tabel migrasi, maka gunakan perintah reset.

php artisan migrate:reset



9. Seeding

Selain fitur *migrations* yang dapat membuat struktur tabel dan relasi menggunakan kode program, kita juga dapat menggunakan fitur *seeding* untuk memasukan data ke tabel tersebut.

Seed adalah segala sesuatu yang harus dimuat dalam sebuah aplikasi untuk memastikan aplikasi dapat berjala dengan baik. Hal ini biasanya dianggap sebagai tahap untuk pengujian sistem dan demo. Sebagian besar aplikasi membutuhkan data referensi untuk dimuat guna untuk memastikan kesuksesan proses pengembangan, testing maupun produksi.

Untuk mempermudah pemahaman tentang *seeding* maka akan kita praktikan langsung. Sebelumnya file migrasi yang telah dibuat di bab sebelumnya, kita migrasikan kembali ke database untuk menghasilkan tabel *users*.

```
php artisan migrate
```

9.1 Membuat Model

Pertama kita buat terlebih dahulu model *Users* yang akan kita letakan di direktori **app/models**. Buat sebuah kelas *Users.php* didalam direktori tersebut kemudian tuliskan sintak berikut.

app/models/Users.php

```
<?php

class Users extends \Eloquent {
    protected $fillable = [];
}
```

9.2 Memulai Seeding

Buat kelas / file php pada direktori **app/database/seeds** dengan nama **UsersTableSeeder.php**

app/database/seeds/UsersTableSeeder.php

```
<?php

class UsersTableSeeder extends Seeder {
    public function run()
    {
        // kosongkan data tabel Users
        DB::table('users')->delete();

        // buat satu buah data users
        Users::create(array(
            'nama' => 'Admin',
            'email' => 'admin@app.com',
            'password' => Hash::make('admin123'),
        ));
    }
}
```

Pada kode diatas ada kode `Hash::make('admin123')`, kode tersebut untuk mengenkripsi password (admin123). Hasil enkripsi nya bisa kita lihat didepan pada saat file *seeder* sudah dieksekusi.

Sesudah kita membuat kelas **UsersTableSeeder.php**, supaya laravel mengenali file *seeder* yang sudah dibuat kemudian kita daftarkan ke kelas **DatabaseSeeder.php** yang berada dalam satu direktori dengan kelas tersebut yaitu **app/database/seeds**.

app/database/seeds/DatabaseSeeder.php

```
<?php

class DatabaseSeeder extends Seeder {

    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Eloquent::unguard();

        $this->call('UsersTableSeederTableSeeder');
    }

}
```

9.3 Menjalankan Seeding

Untuk menjalankan file *seeder*, masih di command prompt gunakan perintah berikut, jika ada konfirmasi untuk melakukan *seeding* ketik (y) lalu tekan Enter.

```
php artisan db:seed
```

```
C:\xampp\htdocs\laravel>php artisan db:seed
*****
*      Application In Production!      *
*****
Do you really wish to run this command? y
Seeded: UsersTableSeederTableSeeder
```

Untuk mengeceknya, masuke ke database **db_laravel** di MySQL, lalu berikan perintah query

```
select * from users
```

MySQL Query Browser - root@localhost:3306 / db_jaravel

File Edit View Query Script Tools Help

Go back Next Refresh `select * from users` Execute Stop

Transaction Explain Compare SELECT FRONT VIEWER GROUP HAVING ORDER SET CREATE

Resultset 1

	id	nama	email	password	created_at	updated_at
▶	1	Admin	admin@app.com	\$2y\$10\$TjzwQB090...	2014-11-13 07:29:05	2014-11-13 07:29:05

1 row fetched in 0.0269s (0.0007s) Edit Apply Changes Discard Changes First Last Search

1: 1

Schemata Bookmarks History

- cdcol
- db_jaravel
 - migrations
 - users
- information_schema
- mysql
- performance_schema

Syntax Functions Params Trx

- Data Manipulation
- Data Definition
- MySQL Utility
- Transactional and Locking

TENTANG PENULIS



Penulis bernama **Teten Nugraha**. Lahir dikota Garut tanggal 18 Desember 1990. Dan Menyelesaikan Pendidikan S1 nya di Universitas Komputer Indonesia.

Saat ini menjabat sebagai Developer dalam proyek *Combat Mangement System* di salah satu perusahaan di Bandung.

Disamping itu menjabat sebagai Owner di **Mapple IT Consult** yang merupakan perusahaan yang bergerak dibidang teknologi informasi dan komunikasi.

Berikut adalah daftar proyek yang telah / sedang ditangani :

- Implementasi Algoritma RSA dalam aplikasi Chat menggunakan Java-RMI
- Sistem Manajemen Beras di PT. Bulog
- Sistem Penjualan di RUG Production
- *Knowledge Management System* di Akademi Angkatan Udara – Yogyakarta
- Sistem Inventori di CV. Dasentek Bandung
- *Combat Management System* (PT. Len – TNI AL)
- Toko Online RealInc with Heru Rusdianto
- Aplikasi Web Report
- Dan lain-lain

Penulis dapat dihubungi di :

- tennugraha777@gmail.com