

VERSI 2.0
JANUARI, 2020



PEMROGRAMAN WEB

DASAR FRAMEWORK LARAVEL – MODUL 6

TIM PENYUSUN: AMINUDIN, S.KOM., M.CS
-IHZA AHMAD ABROR AMRULLAH

PRESENTED BY: LAB. TEKNIK INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG

PEMROGRAMAN WEB

CAPAIAN PEMBELAJARAN MATA KULIAH

1. Mahasiswa mampu menjelaskan dan menerapkan lingkungan yang terdapat di dalam Framework Laravel.
2. Mahasiswa mampu mengimplementasikan Framework Laravel ke dalam pembuatan aplikasi sederhana.

SUB CAPAIAN PEMBELAJARAN MATA KULIAH

1. Mahasiswa mampu menerapkan penggunaan library menggunakan Composer yang terdapat di packagist.org.
2. Mahasiswa mampu melakukan instalasi dan konfigurasi Framework Laravel.
3. Mahasiswa mampu membangun aplikasi sederhana dengan menggunakan Framework Laravel.
4. Mahasiswa mampu memahami konsep MVC di dalam Framework Laravel ke dalam pembuatan aplikasi.

KEBUTUHAN HARDWARE & SOFTWARE

Hardware dan Infrastruktur:

- Laptop/ PC
- Koneksi Internet

Software

- Text Editor (Atom/Sublime/Notepad++/atau lainnya)
- XAMPP (Web Server, MySql, PHP)
- Composer

MATERI POKOK

1. Composer

a. Pengertian Composer

Dalam beberapa tahun belakangan, terjadi perubahan yang sangat signifikan dalam dunia pemrograman php. Yakni munculnya dependencies manager yang bernama composer. Composer adalah sebuah project open source yang dimotori oleh Nils Adermann dan Jordi Boggiano. Project composer ini dihost di github (<https://github.com/composer/composer>) tercatat sejak tanggal 3 April 2011 dan masih aktif sampai sekarang. Composer dikenal jugasebagai dependency management maksudnya dengan menggunakan composer developerdapat menginstall suatu library melalui composer dan composer akan secara otomatismenginstall library lain yang dibutuhkan, tanpa perlu mendownload satu persatu. Mirip denganapt get install di sistem operasi linux, npm yang ada di dalam NodeJs dan Bundler pada Ruby. Berikut salah satu contoh penggunaan Composer di dalam pemanggilan suatu library :

- Kita punya project yang butuh library. Misalnya, library untuk ekspor data ke Excel yaitu PHPEXcel dan DomPDF buat ekspor data ke PDF.
- Library-library tersebut juga butuh library lainnya. Dalam kasus ini, PHPEXcel butuh libraryext-xml dan DomPDF butuh phenx/php-font-lib. Kasus seperti ini, namanya dependensi. Jadi, bisa disebut PHPEXcel memiliki dependensi ext-xml.
- Karena memang banyak dependency-nya maka, dengan menggunakan composer libraryyang mempunyai dependency akan secara otomatis diinstall.

Jadi, cara kerja dari composer yaitu kita menuliskan library yang dibutuhkan di dalam suatu aplikasi kemudian composer akan mencari versi yang sesuai lalu install secara otomatis berikut dependency dari library tersebut. Dengan composer, developer tidak repot download library PHP manual satu-persatu. Begitupun dengan proses update library, tidak perlu diupdate satu-persatu, cukup diubah satu file (composer.json), jalankan perintah composer update lalu mengupdate semua library. Cara mempraktekan penggunaan library di dalam composer akan dipraktekan pada Sub-Bab Lembar Kerja.

b. Cara Kerja Composer

Cara kerja composer cukup sederhana, nantinya Anda diharuskan untuk menulis library yang dibutuhkan di sebuah file (composer.json). Kemudian composer akan mencari dan meng-install library tersebut secara otomatis. Begitu pula jika terdapat update terbaru pada library yang digunakan, cukup ubah file composer.json kemudian secara otomatis pula composer akan meng-updatenya. Pada intinya composer secara tidak langsung membantu developer dalam meringankan pekerjaannya, karena composer melakukan install maupun update library secara otomatis.

Sebelum men-download library-library yang kita perlukan, Composer akan:

1. Mengecek apakah library-library ini memerlukan library lain sebagai dependensinya?
2. Mengecek versi PHP dan module – module PHP yang diperlukan oleh library – library tersebut.

Setelah kedua proses di atas, Composer mendownload library – library tersebut ke dalam aplikasi / project kita pada direktori “vendor”. Misalkan versi PHP yang terinstall di PC kita tidak cocok dengan versi minimum yang dibutuhkan oleh salah satu library, maka Composer akan memberitahukan bahwa “spesifikasi minimum tidak terpenuhi”. Atau ketika ada module PHP yang diperlukan oleh salah satu library belum terinstall atau enabled, Composer juga akan memberitahukan dan menyarankan kita untuk menginstall/enable module terlebih dahulu. Jadi selain mengecek dependensi library, Composer juga membantu mengecek spesifikasi PHP kita apakah cocok atau tidak dengan kebutuhan library tersebut.

2. Framework Laravel

Framework laravel merupakan salah satu framework PHP berbasis MVC yang digunakan untuk membangun aplikasi web skala kecil maupun skala besar. Framework ini dirilis oleh MIT dan disediakan secara gratis di github. Framework ini dibangun pertama kali oleh Pencetusnya Martin Otwell pada tahun 2011. Meskipun laravel dapat dikatakan masih relatif baru jika dibandingkan dengan para pendahulunya yang sudah terkenal dikalangan developer web sebut saja CI (Code Iginiter), Symfoni, Ruby on Rails, YII dsb. Tetapi laravel mampu bersaing dikalangan persaingan pembuatan aplikasi web menggunakan framework PHP dan telah menjadi salah satu primadona framework laravel pada perusahaan start-up IT di Indonesia.

3. Kosnep MVC Framework Laravel

Dicuplik dari Wikipedia konsep Model-View-Controller atau MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (Model) dari tampilan (View) dan cara

bagaimana memprosesnya (Controller). Dalam implementasinya kebanyakan kerangka kerja (framework) dalam aplikasi web adalah berbasis arsitektur MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi web.

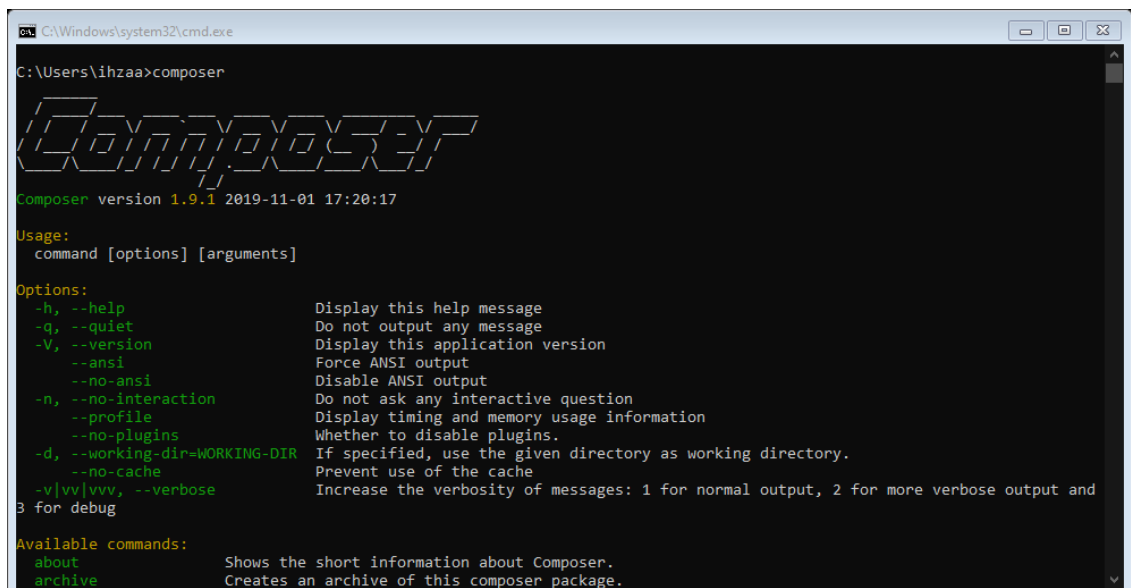
Munculnya konsep MVC inilah yang mendasari lahirnya framework yang elegan, rata-rata framework yang muncul saat ini basisnya OOP dan MVC. Intinya di dalam konsep MVC ini ada semacam pemisahan code yang berhubungan dengan aplikasi. Misalnya saja code yang berhubungan dengan data dimasukkan di dalam konsep Model sedangkan code yang berhubungan dengan logic aplikasi dihubungkan dengan konsep Controller lalu code yang berhubungan dengan View semisal HTML, CSS dan Javascript masuk ke dalam konsep View. Lalu, dibenak kita ada pertanyaan apa untungnya pemisahan code seperti itu, jika kita bayangkan kalau kita mengerjakan proyek yang bisa dikatakan sebagai big-project bagaimana susahya apabila semua code yang berhubungan dengan aplikasi ditampilkan dalam satu file. Tetapi dengan adanya pemisahan code ini berakibat dapat membagi mana yang resource yang berhubungan dengan logic, view dan data dapat dipisahkan menurut dari konsep tersebut. Di dalam Framework Laravel konsep MVC sudah diterapkan secara baik di tiap pengelompokanya. Meskipun developer yang baru belajar Laravel pasti dapat menguasai konsep MVC yang ada di dalam Laravel.

LEMBAR KERJA

1. Composer

a. Instalasi Composer

Cara instalasi composer dapat diakses melalui link berikut <https://www.malasngoding.com/cara-install-composer/>. Setelah selesai menginstall composer, buka command prompt lalu ketikan 'composer', jika sudah terinstall dengan benar maka composer akan menampilkan seperti pada gambar berikut :



```

C:\Windows\system32\cmd.exe
C:\Users\ihzaa>composer

Composer version 1.9.1 2019-11-01 17:20:17

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version              Display this application version
      --ansi                Force ANSI output
      --no-ansi             Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
      --profile              Display timing and memory usage information
      --no-plugins           Whether to disable plugins.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
      --no-cache             Prevent use of the cache
  -v|vv|vvv, --verbose      Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
                             3 for debug

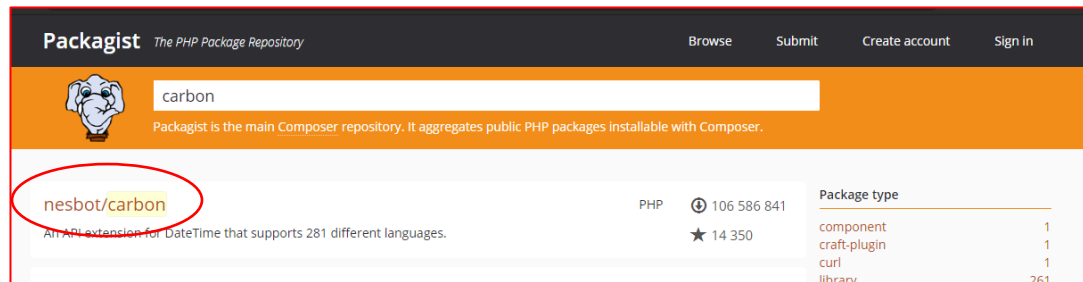
Available commands:
  about      Shows the short information about Composer.
  archive    Creates an archive of this composer package.
  
```

b. Cara Menggunakan Library di Composer

Untuk mempelajari bagaimana cara menggunakan composer ini berdasarkan dari teori yang sudah dibahas diatas, maka di dalam lembar kerja ini akan dicontohkan sebuah kasus penggunaan

composer. Library yang akan digunakan adalah library Carbon. Library ini digunakan untuk memanipulasi tampilan objek DateTime. Pernah melihat tampilan tanggal “2 hour ago”, “2 minutes ago” atau “A month ago” di web? Itu salah satu fungsi dari Carbon. Untuk mendapatkan package dengan Composer,

1. Pertama kita kunjungi ke pusat package, kunjungi di <https://packagist.org/>. Setelah terbuka, ketikkan “carbon” di kotak pencarian dan klik pada hasil pertama yang muncul yaitu “nesbot/carbon”.



2. Untuk install package carbon tersebut, buat folder di dalam htdocs misal dengan nama **contohComposer**. Di dalam folder **contohComposer** tersebut, buatlah file baru bernama **composer.json** kemudian isikan script berikut :

```
{
  "require" : {
    "nesbot/carbon" : "^2.16"
  }
}
```

Script tersebut digunakan untuk memanggil package carbon dengan versi yang telah di cantumkan.

3. Selanjutnya, untuk mendownload package tersebut, masuk ke command prompt arahkan ke folder **contohComposer** yang berada di dalam folder htdocs lalu ketikan perintah **composer install**.

```
C:\xampp\htdocs\contohComposer>composer install
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 4 installs, 0 updates, 0 removals
- Installing symfony/translation-contracts (v2.0.1): Loading from cache
- Installing symfony/polyfill-mbstring (v1.14.0): Downloading (100%)
- Installing symfony/translation (v5.0.4): Downloading (100%)
- Installing nesbot/carbon (2.30.0): Downloading (100%)
symfony/translation suggests installing symfony/config
symfony/translation suggests installing symfony/yaml
symfony/translation suggests installing psr/log-implementation (To use logging capability in translator)
Writing lock file
Generating autoload files
C:\xampp\htdocs\contohComposer>
```

4. Apabila sudah benar dan selesai cara instalasinya maka, di dalam folder **contohComposer** akan terbentuk beberapa file dan folder baru. Berikut penjelasan file dalam folder itu:
 - **composer.json**, ini file yang kita buat, berisi dependensi library dari project kita.
 - **composer.lock**, file ini mencatat versi package yang saat ini terinstal.
 - **vendor**, folder ini berisi package yang telah kita install. Setiap package yang kita tulis di bagian require akan di download ke folder ini.

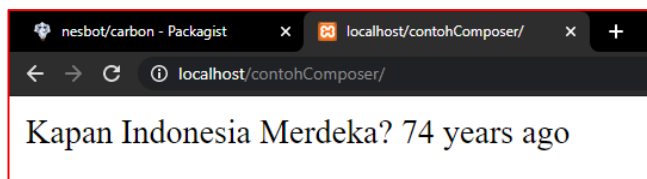
- **vendor/autoload.php**, berfungsi memanggil autoloader dari composer.

Salah satu keunggulan dari composer adalah autoloader. Fitur ini berfungsi untuk memanggil class yang sesuai ketika kita membutuhkan class dari suatu library. Jadi, jika banyak library yang kita install kita hanya cukup menulis require untuk file vendor/autoload.php. Dan, autoload ini cukup cerdas dengan hanya me-load class yang kita butuhkan.

5. Untuk mempraktekan penggunaan package Carbon, buat file dengan **index.php** di dalam folder **contohComposer** kemudian isikan script berikut :

```
<?php
require 'vendor/autoload.php';
use Carbon\Carbon;
date_default_timezone_set('Asia/Jakarta');
$date = Carbon::parse('1945-8-17');
printf("Kapan Indonesia Merdeka? %s\n",$date->diffForHumans());
?>
```

6. Jalankan index.php tersebut melalui web browser, menggunakan url <http://localhost/contohComposer/>.



7. Ketika developer akan menambahkan library, misalnya library yang ditambahkan berhubungan dengan text mining dalam hal ini untuk mengolah kata, menghilangkan imbuhan biasanya disebut juga dengan stemming, packagist.org telah menyediakan package tersebut dengan nama **sastrawi**. Misalnya, kita akan menambahkan menambahkan library tersebut, edit **composer.json** kemudian tambahkan package sastrawi.

```
{
  "require": {
    "nesbot/carbon": "^2.16",
    "sastrawi/sastrawi": "^1"
  }
}
```

8. Kemudian command prompt, dan diingat masih dalam direktori **contohComposer** ketikkan perintah `composer update`.

```
C:\xampp\htdocs\contohComposer>composer update
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Installing sastrawi/sastrawi (v1.2.0): Downloading (100%)
Writing lock file
Generating autoload files
```

9. Untuk menerapkan package sastrawi ke dalam suatu program, Buat file baru dengan nama **index2.php**, di dalam directory **contohComposer**, lalu isikan script berikut:

```
<?php
require 'vendor/autoload.php';

$stemmerFactory = new Sastrawi\Stemmer\StemmerFactory();
$stemmer = $stemmerFactory->createStemmer();

$sentence = 'Perekonomian Indonesia sedang dalam pertumbuhan yang
membangggakan';
$output = $stemmer->stem($sentence);
echo $output . "\n";
?>
```

10. Jalankan index2.php melalui web browser dengan url <http://localhost/contohComposer/index2.php>.



2. Instalasi dan Konfigurasi Laravel

a. Kebutuhan Instalasi Laravel

Agar versi terbaru Laravel dapat diinstal di computer local, dibutuhkan beberapa requirement sebagai berikut :

- PHP >= 7.2.0
- BCMath PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

Paling penting, untuk instalasi Laravel versi yang terbaru harus perhatikan versi PHP yang telah terinstall pada computer sesuai dengan requirement minimal Laravel. Untuk mengecek versi PHP yang telah diinstall dapat di cek melalui url <http://localhost/dashboard/phpinfo.php>.

PHP Version 7.3.9



b. Instalasi dan Konfigurasi Laravel

1. Untuk Melakukan instalasi Laravel dapat dilakukan menggunakan command prompt yang terlebih dahulu install composer. Silahkan masuk ke directory htdocs kemudian ketikan perintah berikut :

```
composer create-project --prefer-dist laravel/laravel
WebLaravel
```

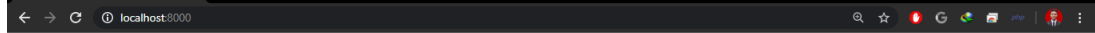
Secara otomatis di dalam directory htdocs akan terbuat satu project atau folder dengan nama WebLaravel beserta dengan dependency nya.

```
C:\Windows\system32\cmd.exe
C:\Users\ihzaa>cd C:\xampp\htdocs

C:\xampp\htdocs>composer create-project --prefer-dist laravel/laravel WebLaravel
Installing laravel/laravel (v6.12.0)
- Installing laravel/laravel (v6.12.0): Loading from cache
Created project in WebLaravel
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 85 installs, 0 updates, 0 removals
- Installing symfony/polyfill-ctype (v1.14.0): Downloading (100%)
- Installing phpoption/phpoption (1.7.2): Loading from cache
- Installing vlucas/phpdotenv (v3.6.0): Loading from cache
- Installing symfony/css-selector (v5.0.4): Loading from cache
- Installing tijsverkoyen/css-to-inline-styles (2.2.2): Loading from cache
- Installing symfony/polyfill-php72 (v1.14.0): Downloading (100%)
- Installing symfony/polyfill-mbstring (v1.14.0): Loading from cache
- Installing symfony/var-dumper (v4.4.4): Loading from cache
- Installing symfony/routing (v4.4.4): Loading from cache
- Installing symfony/process (v4.4.4): Loading from cache
- Installing psr/log (1.1.2): Loading from cache
- Installing symfony/polyfill-php73 (v1.14.0): Downloading (100%)
- Installing symfony/polyfill-intl-idn (v1.14.0): Downloading (100%)
- Installing symfony/mime (v5.0.4): Loading from cache
- Installing symfony/http-foundation (v4.4.4): Loading from cache
- Installing symfony/event-dispatcher-contracts (v1.1.7): Loading from cache
- Installing psr/container (1.0.0): Loading from cache
- Installing symfony/event-dispatcher (v4.4.4): Loading from cache
- Installing symfony/debug (v4.4.4): Loading from cache
- Installing symfony/error-handler (v4.4.4): Loading from cache
```

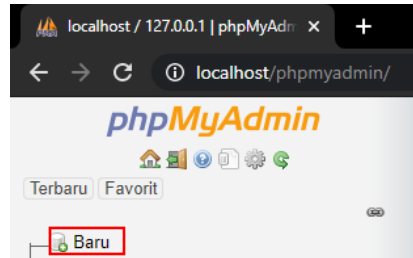
2. Masih pada command prompt, pindah ke direktori WebLaravel yang baru terbuat dengan perintah :
`cd WebLaravel`
 Setelah command prompt berada di direktori WebLaravel ketikkan perintah:
`php artisan serve`

3. Buka web browser kemudian jalankan : <http://localhost:8000/>



4. Directory WebLaravel yang sudah terbentuk pada langkah nomer 1, untuk mengetahui fungsi dari masing-masing directory dapat diakses melalui alamat web resmi Laravel berikut <https://laravel.com/docs/6.x/structure>.
5. Perlu diperhatikan bahwa ada beberapa directory yang akan sering digunakan atau diedit di dalam membuat aplikasi menggunakan Laravel, diantaranya :
- routes/web.php** directory ini digunakan untuk mengarahkan suatu resource tertentu, untuk cara penulisan dapat diakses di alamat berikut <https://laravel.com/docs/6.x/routing>
 - resource/views** direktori ini khusus untuk menangani tampilan di sisi user (frontend) cara pembuatan view dapat di akses di halaman berikut <https://laravel.com/docs/6.x/views>
 - app/Http/Controllers** directory ini untuk menangani proses logic dan menghubungkan antara model dengan views, adapun cara pembuatan controller dapat diakses di halaman berikut <https://laravel.com/docs/6.x/controllers>
 - app** directory ini digunakan untuk file model, di mana file model berhubungan dengan scripting data, cara pembuatan model dapat diakses di halaman berikut <https://laravel.com/docs/6.x/eloquent>
- c. **Contoh Aplikasi Sederhana Menggunakan MVC di dalam Laravel**
- Di dalam contoh aplikasi inim akan dipraktekan cara menampilkan data dari database dengan menggunakan konsep MVC yang ada di dalam Framework Laravel. Untuk itu mohon diikuti langkah-langkah sebagai berikut :
1. **Pembuatan Database**
- Buat database melalui phpMyAdmin atau MySQL, contoh database yang dibuat dengan nama 'dbWebLaravel'. Caranya:
- Buka <http://localhost/phpmyadmin/>

b) Tekan tombol new atau baru



c) Isi nama database, dalam contoh ini 'dbWebLaravel' , lalu tekan buat/create.



2. Buka file .env

File tersebut digunakan untuk menghubungkan antara database dengan Laravel, kemudian ganti bagian tersebut sesuai dengan settingan database yang telah dibuat

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dbWebLaravel
DB_USERNAME=root
DB_PASSWORD=
```

Secara default isian parameter database di file .env (DB_HOST, DB_DATABASE, DB_USERNAME, DB_PASSWORD) akan sesuai dengan credential database MySQL dan PostgreSQL yang telah terinstall. Kita juga dapat mengecek, koneksi antara database dengan Laravel dengan cara masuk ke folder project, kemudian menggunakan tinker untuk menjalankan beberapa perintah pengecekan database sudah bisa digunakan atau tidak. Ketikkan perintah `php artisan tinker` pada command prompt masih pada direktori WebLaravel.

```
C:\xampp\htdocs\WebLaravel>php artisan tinker
Psy Shell v0.9.12 (PHP 7.3.9 - cli) by Justin Hileman
>>> DB::Connection()
=> Illuminate\Database\MySQLConnection {#3008}
>>>
```

3. Membuat Migrate

Untuk membuat table baru dengan migration, kita harus membuat file migration terlebih dahulu dengan perintah :

```
php artisan make:migration --create=nama_table nama_file
```

Mari kita buat migration untuk membuat table berita :

```
php artisan make:migration --create=berita create_table_berita
```

Secara otomatis maka di dalam directory database/migration akan ter-create satu file dengan nama **create_table_post**. Buka file **create_table_post.php**, file tersebut digunakan untuk membuat table dengan menggunakan script PHP, ini yang dinamakan dengan migration. Kemudian edit dibagian fungsi up dengan menambahkan 3 buah field yang akan di tambahkan

```
public function up()
{
    Schema::create('berita', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('judul');
        $table->longText('isi');
        $table->enum('published', ['yes', 'no']);
        $table->timestamps();
    });
}
```

Ada sangat banyak method untuk membuat field yang didukung oleh Blueprint, karena pada method up kita membuat tabel baru, maka pada method down kita harus menghapus table tersebut menggunakan method Schema::drop() dengan parameter nama table yang akan kita hapus. Untuk menjalankan perintah artisan kita dapat menggunakan perintah php artisan migrate. Kemudian lihat di <http://localhost/phpmyadmin/>.

Ketika menjalankan perintah php artisan migrate. **Jika** kemudian muncul error **Illuminate\Database\QueryException** maka cara penanganannya adalah :

- Buka file **app/Providers/AppServiceProvider.php**, kemudian edit method boot. Tambahkan script yang di bold berikut :

```
<?php

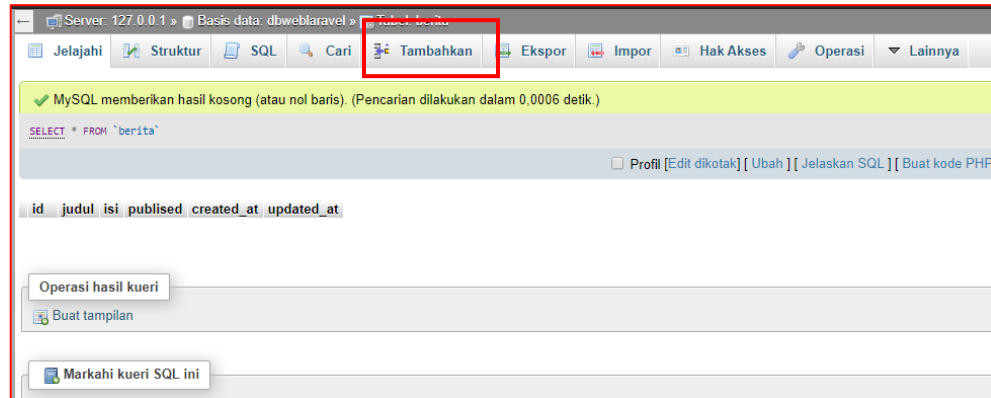
namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;

class AppServiceProvider extends ServiceProvider
{
    .....
    public function boot()
    {
        Schema::defaultStringLength(191);
    }
}
```

- Jalankan kembali perintah php artisan migrate
- Kalau sudah sukses, sekarang lihat database WebLaravel, maka akan muncul beberapa table baru di dalamnya

- Masukkan atau tambahkan beberapa data ke dalam table berita secara manual. Dengan cara pilih atau klik table berita -> pilih atau klik tombol tambahkan atau add.



4. Membuat Model

Membuat model untuk mengatur penggunaan data di dalam Laravel, dengan cara ketikkan perintah di command prompt pada directory WebLaravel

```
php artisan make:model berita
```

Maka secara otomatis akan terbentuk satu file dengan nama **berita.php** di directory **\app**. Secara default model akan terkoneksi dengan table dengan nama yang sama dengan nama model tetapi berimbuhan **s**. Misal jika membuat model bernama **berita** maka otomatis akan terkoneksi dengan table bernama **beritas**, karena pada contoh migrasi di atas kita membuat table bernama **berita** maka kita harus mengkonfigurasi file **berita.php**.

Buka file **berita.php** pada directory **\app**, tambahkan kode berikut :

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class berita extends Model
{
    protected $table = 'berita'; #ini baris yang ditambahkan
}
```

5. Membuat Routing

Buka file **web.php** pada directory **routes/web.php**, lalu tambahkan kode berikut :

```
Route::get('berita', 'beritaController@index');
```

6. Membuat Controller

Untuk membuat controller di dalam project ini, seperti biasanya masuk ke command prompt di dalam directory WebLaravel, ketikkan perintah :

```
php artisan make:controller beritaController --resource
```

Maka secara otomatis di dalam directory **WebLaravel/app/Http/Controllers**, akan terbentuk satu file dengan nama **beritaController.php**, lalu buka file tersebut dan perhatikan

secara otomatis akan ada beberapa fungsi diantaranya `index()`, `create()`, `store()` dll, dan tambahkan kode berikut di dalam fungsi `index()` :

```
public function index()
{
    return view('index');
}
```

7. Membuat View

- Untuk menampilkan data di dalam Laravel, maka harus ditampilkan lewat views yang berada di dalam directory **resources/views** kemudian buat satu file di dalam folder tersebut dengan nama **master.blade.php**, file ini digunakan untuk menampung keseluruhan file yang ada di dalam views. Isikan file **master.blade.php** dengan kode berikut :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    {{-- TITLE UNTUK HALAMAN --}}
    <title>@yield('title')</title>
</head>
<body>
    {{-- KONTEN HALAMAN --}}
    @yield('body')
</body>
</html>
```

- Untuk menampilkan data yang sudah tersimpan di database buat satu file dengan nama **index.blade.php** yang berada di dalam folder **resources/views**.

```
@extends('master')
@section('title', 'Halaman Utama Portal - Kabar Burung')
@section('body')
<h1>Portal - Kabar Burung</h1>
<table border="1">
    <thead>
        <tr>
            <th>No</th>
            <th>Judul</th>
            <th>Published</th>
            <th>Tanggal</th>
        </tr>
```

```

</thead>
<tbody>
    <?php $no=1 ?>
    @foreach (App\berita::all() as $berita)
    <tr>
        <td>{{ $no++ }}</td>
        <td>{{ $berita->judul }}</td>
        <td>{{ $berita->published }}</td>
        <td>{{ $berita->created_at->format('M d, Y') }}</td>
    </tr>
    @endforeach
</tbody>
</table>
@endsection

```

8. Hasil Program

Jalankan dengan menggunakan perintah `php artisan serve` pada command prompt dan pada web browser buka alamat <http://localhost:8000/berita>



9. Membuat Fitur Create atau tambah Berita

- Buat route baru pada file **routes/web.php**:

```

Route::get('berita/tambah', 'beritaController@create');
Route::post('berita/tambah/store', 'beritaController@store');

```

- Pada file **beritaController.php** tambahkan kode berikut yang bergaris bawah:

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

```

```

class beritaController extends Controller
{
    . . .
    public function create()
    {
        return view('tambah');
    }

    public function store(Request $request)
    {
        date default timezone set('Asia/Jakarta');

        DB::table('berita')->insert([
            'judul' => $request->judul,
            'isi' => $request->isi,
            'published' => $request->published,
            'created at' => date('Y-m-d h:i:s', time())
        ]);
        return redirect('/berita');
    }
    . . .

```

- Ubah file **index.blade.php** menjadi seperti berikut :

```

@extends('master')
@section('title', 'Halaman Utama Portal - Kabar Burung')
@section('body')
<h1>Portal - Kabar Burung</h1>
<a href="/berita/tambah">Tambah Berita</a>
<table border="1">
    <thead>
        <tr>
            <th>No</th>
            <th>Judul</th>
            <th>Published</th>
            <th>Tanggal</th>
        </tr>
    </thead>
    <tbody>
        <?php $no=1 ?>
        @foreach (App\berita::all() as $berita)
        <tr>
            <td>{{ $no++ }}</td>
            <td>{{ $berita->judul }}</td>
            <td>{{ $berita->published }}</td>

```

```

        <td>{{ $berita->created_at->format('M d, Y') }}</td>
    </tr>
    @endforeach
</tbody>
</table>
@endsection

```

- Pada directory **resources/views/** tambahkan file **tambah.blade.php**. Lalu isikan kode berikut pada file **tambah.blade.php** :

```

@extends('master')
@section('title', 'Halaman Utama Portal - Kabar Burung')
@section('body')
<h1>Halaman Tambah Berita</h1>
<form action="/berita/tambah/store" method="POST">
    @csrf
    <label for="judul">Judul : </label>
    <input type="text" id="judul" name="judul">
    <br>
    <br>
    <label for="isi">Isi : </label>
    <textarea name="isi" id="isi" rows="5"></textarea>
    <br>
    <br>
    <label for="published">Published : </label>
    <select name="published" id="published">
        <option value="yes">Yes</option>
        <option value="no">No</option>
    </select>
    <br>
    <br>
    <button type="submit">Tambahkan</button>
</form>
@endsection

```

- Pada command prompt jalankan kembali `php artisan serve` , lalu buka kembali url <http://localhost:8000/berita>, anda sudah dapat menambahkan berita langsung melalui halaman website yang sudah dibuat diatas.

TUGAS PRAKTIKUM

KEGIATAN 1

Berdasarkan dari Sub-Bab Lembar Kerja No.2 dan berdasarkan dari aplikasi yang telah dibuat, pada halaman url <http://localhost:8000/berita>, ubah isi tampilan kolom **Tanggal** menjadi seperti “10 seconds ago”, “2 hours ago” dan seterusnya.



KEGIATAN 2

Berdasarkan dari Sub-Bab Lembar Kerja No.2 dan berdasarkan dari aplikasi yang telah dibuat, kembangkan aplikasi menjadi :

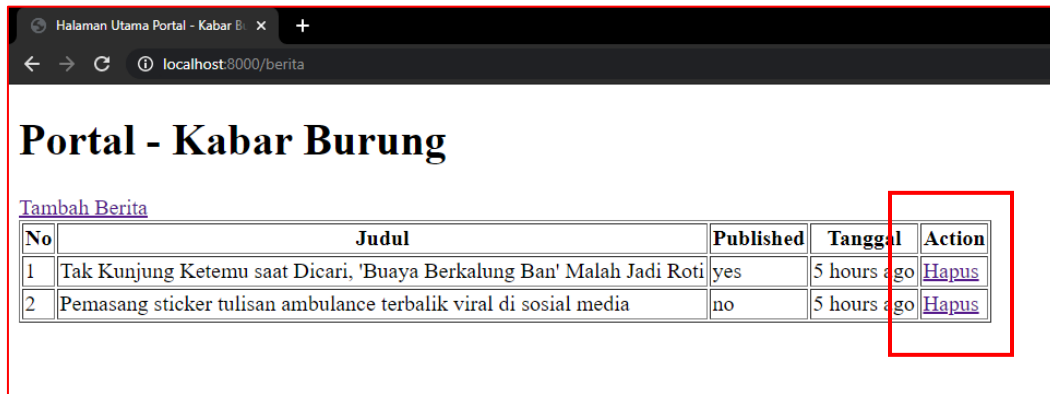
1. Ketika user memasukkan alamat <http://localhost:8000/> maka akan menampilkan data berita, seperti contoh pada gambar dibawah.
2. Berita yang ditampilkan pada alamat <http://localhost:8000/> hanyalah berita yang berstatus published 'yes'.



KEGIATAN 3

Berdasarkan dari Sub-Bab Lembar Kerja No.2 dan berdasarkan dari aplikasi yang telah dibuat, kembangkan aplikasi menjadi :

1. Pada halaman <http://localhost:8000/berita>, terdapat 1 kolom tambahan di masing – masing baris berita yang berisi tombol untuk menghapus berita pada baris tersebut dari database dan akan me-refresh halaman <http://localhost:8000/berita> guna memastikan berita telah terhapus.



No	Judul	Published	Tanggal	Action
1	Tak Kunjung Ketemu saat Dicari, 'Buaya Berkalung Ban' Malah Jadi Roti	yes	5 hours ago	Hapus
2	Pemasang sticker tulisan ambulance terbalik viral di sosial media	no	5 hours ago	Hapus

RUBRIK PENILAIAN

1. Mengerjakan dan mampu menjelaskan kegiatan 1. (Point Max 30)
2. Mengerjakan dan mampu menjelaskan kegiatan 2. (Point Max 35)
3. Mengerjakan dan mampu menjelaskan kegiatan 3. (Point Max 35)