Laravel from Scratch

Arief Setya
Rekayasa Perangkat Lunak
Sekolah Menengah Kejuruan Negeri 10 Jakarta

Laravel, adalah salah satu dari banyaknya free and open source framework yang banyak digunakan para Web Developer untuk membuat projectnya. Framework ini dibuat oleh Taylor Otwell dan ditujukan untuk pengembang aplikasi Web yang menggunakan pola MVC (Model-View-Controller). Laravel sendiri mempunyai fitur yang diutamakan untuk kemudahan penggunaanya yaitu syntax yang ekspresif, sistem package dengan dependency manager, cara yang berbeda untuk mengakses database dan berbagai utilitas yang membantu dalam pengembangan aplikasi dan pemeliharaan.

Merujuk pada survey yang dilakukan pada Maret 2015 dalam framework PHP yang populer, Laravel menempati urutan pertama, diikuti Symfony2, Nette, CodeIgniter, Yii2 dan yang lainnya. Seperti pada Agustus 2014, Laravel adalah yang paling populer dan PHP project yang paling banyak dilihat di GitHub.

Laravel sendiri dirilis dibawah MIT License, dengan kode sumber yang disimpan di GitHub.

Sejarah Laravel

Laravel 1

Laravel dibuat oleh Taylor Otwell dengan fitur yang lebih banyak sebagai alternatif CodeIgniter yang tidak memberikan fitur canggih seperti autentikasi dan otorisasi. Versi beta dirilis pada 9 Juni 2011, diikuti dengan Laravel 1 yang dirilis selanjutnya pada bulan yang sama. Laravel 1 dilengkapi dengan fitur autentikasi, lokalisasi bahasa, model, view, session, routing dan mekanisme lainnya tapi belum didukung adanya controller sehingga belum dapat dikatakan sebagai MVC pada umumnya.

Laravel 2

Laravel 2 dirilis pada September 2011, dengan dilengkapi beberapa improvisasi oleh pembuat dan komunitasnya. Fitur utamanya yaitu adanya controller, sehingga membuat framework ini murni Framework MVC, kemudian adanya dukungan untuk Inversion of Control (IoC), dan sistem template yang disebut "blade". Sebaliknya, dukungan untuk pihak ketiga dihapus pada Laravel 2.

Laravel 3

Laravel 3 dirilis pada Februari 2012 dengan fitur baru antara lain Command-line interface (CLI) yang disebut "Artisan", dukungan lebih untuk sistem manajemen database, version control untuk database yang disebut "migrations", dukungan untuk penanganan event dan packaging system yang disebut "bundles"

Laravel 4

Laravel 4 dengan nama kode "Illuminate" dirilis pada Mei 2013. Pada versi ini laravel dibentuk sebagai package dan didistribusikan melalui Composer, dukungan untuk Database seeding, message queues, dukungan untuk pengiriman email yang berbeda tipe, dukungan untuk penghapusan data di database dengan sistem tunggu yang disebut "soft deletion".

Laravel 5

Laravel 5 dirilis pada Februari 2015, perubahan yang paling signifikan adalah struktur direktori baru yang banyak berubah dari yang sebelumnya pada Laravel 4.3. Package terbaru antara lain adalah scheduler, flysystem, elixir dan socialite. Versi paling baru saat ini adalah Laravel 5.1 yang dirilis pada Juni 2015, yang menerima dukungan jangka panjang (LTS).

Untuk versi Laravel yang akan kita bahas kali ini adalah **Laravel 5.0** dan sistem operasi untuk latihan adalah **Linux Ubuntu 14.04 LTS**.

Pengenalan

Struktur direktori

```
> app
  > Commands
  > Console
    > Commands
  > Events
  > Exceptions
  > Handlers
    > Commands
    > Events
  > Http
    > Controllers
    > Middleware
    > Requests
  > Providers
  > Services
> bootstrap
> config
> database
  > migrations
  > seeds
> public
> resources
  > assets
    > less
  > lang
    > en
  > views
> auth
> emails
> errors
> vendor
> storage
  > app
  > framework
  > logs
> tests
> vendor
```

Dari id-laravel.com

app/Http

Direktori ini merupakan direktori yang dibuat secara khusus untuk menyimpan seluruh file-file yang berkaitan dengan proses request dan response Http. Dikretori ini memiliki tiga buah sub direktori yang diantaranya adalah "Controllers", "Middleware" dan "Requests". Berikut ini adalah penjelasan mengenai fungsi dari ketiga buah sub direktori tersebut:

- `app/Http/Controllers` : Direktori ini digunakan untuk menyimpan seluruh class Controller yang kita buat seperti misalnya ProductController.php, SalesController.php, dll.
- `app/Http/Middleware` : Direktori ini digunakan untuk menyimpan seluruh class yang berhubungan dengna middleware PHP. Secara umum middleware adalah sebuah class yang akan dieksekusi sebelum HTTP request yang masuk diberikan kepada Controller. Tujuan dari class Middleware adalah untuk melakukan filter seperti misalnya menolak akses dari user yang belum login.
- `app/Http/Requests` : Direktori ini hanya berisikan sebuah class yaitu Request.php yang dapat digunakan untuk mendapatkan data dari form request yang dikirim oleh web browser. Selain itu direktori ini juga ditujukan untuk menyimpan class validator yang kita buat baik yang dibuat secara manual ataupun dengan menggunakan perintah php artisan `make:request`.

database/migrations

Direktori ini berisikan file-file migrations yang digenerate oleh laravel pada saat kita menjalankan perintah php artisan `make:migration`. fitur migration sendiri sangat berguna untuk melakukan perubahan pada database baik itu penambahan tabel, penambahan kolom, menghapus kolom, menghapus tabel serta melakukan roll-back setiap perubahan database yang kita buat. Fitur migration ini akan sangat terasa manfaatnya terutama pada saat kita mengerjakan sebuah project di dalam sebuah tim dan banyak struktur database yang berubah seiring perkembangan project.

database/seeds

Direktori ini berisikan file-file dabase seeds yang digenerate oleh laravel pada saat kita menjalankan perintah php artisan `make:seeder`. fitur seeding di laravel sendiri sangat berguna apabila kita ingin melakukan inisialisasi data (data awalan) pada table yang kita buat.

public

Pada dasarnya laravel memisahkan antara direktori public dan private. direktori public adalah direktori dimana seluruh resource aplikasi dapat diakses melalui web browser seperti misalnya gambar, javascript dan css. Sedangkan direktori private sendiri berisikan seluruh kode PHP yang telah kita buat ataupun yang merupakan bawaan dari framwork laravel itu sendiri. Umumnya, dalam melakukan proses deployment laravel yang secure, hanya direktori public ini sajalah yang diletakkan di dalam direktori `public_html` pada web server sedangkan direktori lainnya diletakkan di luar direktori `public_html`.

resources

Direktori ini memiliki tiga buah sub direktori yaitu “assets”, “lang” dan views. Berikut ini adalah penjelas singkat terkait fungsi dari masing-masing sub direktori tersebut:

- `assets` : Sejak rilis versi 5, laravel memiliki sebuah fitur yang bernama laravel elixir. Fitur ini ditujukan untuk membantu para pengguna laravel untuk meng-compile file less, saas dan coffescript yang mereka buat. Nah, direktori ini ditujukan untuk menyimpan resources tersebut yang nantinya akan secara otomatis dicompile oleh laravel dengan menggunakan gulp dan dipindahkan ke dalam direktori public. Selain itu kita juga dapat menyimpan resources berupa image atau berkas-berkas lain yang nantinya akan dipindahkan oleh laravel kedalam direktori public dengan cara yang sama.
- `lang` : Secara default laravel sudah memiliki support terhadap implementasi localization yang dapat membantu para pengguna framework untuk menciptakan aplikasi web yang multi bahasa. Direktori ini menyimpan seluruh definisi bahasa yang telah kita buat.
- `views` : Direktori ini digunakan untuk menyimpan seluruh file html atau template blade yang kita buat.

test

Laravel merupakan sebuah framework yang didesain dengan mindset testable framework. Oleh karena itu, secara default laravel sudah menyediakan library-library yang dibutuhkan untuk dapat melakukan unit testing seperti PHPUnit dan Mockery. Nah, direktori ini berfungsi untuk menyimpan seluruh file test yang dibuat untuk kemudian dijalankan oleh PHPUnit.

Sejauh ini sepertinya kita sudah memaparkan beberapa direktori penting yang harus diperhatikan dalam menggunakan laravel 5. Eh, tunggu dulu. Kayaknya masih ada yang ketinggalan nih, kira-kira apa ya?

Dimanakah kita dapat meletakkan class Model di laravel 5?

Salah satu yang membuat kita bingung pada saat menggunakan laravel 5 adalah bahwa tidak adanya direktori bernama "Models" seperti framework-framework PHP MVC lainnya. Lalu, dimanakah seharusnya kita meletakkan class Models yang sudah dibuat? Apakah kita harus membuat sendiri folder bernama "Models" di `app/Http` ? Sebetulnya tidak ada konvensi khusus dimana letak class model berada, akan tetapi jika melihat default model user User.php yang disediakan oleh laravel serta lokasi generated model yang dibuat oleh laravel pada saat kita menggunakan perintah php artisan make:model maka dapat disimpulkan bahwa lokasi class Model pada laravel 5 adalah di dalam folder app seperti misalnya `app\Product.php`.

MVC (Model-View-Controller)

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 framework dalam aplikasi website 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.

Model

Model mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain.

View

View adalah bagian yang mengatur tampilan ke pengguna. Bisa di katakan berupa halaman web.

Controller

Controller merupakan bagian yang menjembatani model dan view. Controller berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman web.

Dengan menggunakan metode MVC maka aplikasi akan lebih mudah untuk dirawat dan dikembangkan. Untuk memahami metode pengembangan aplikasi menggunakan MVC diperlukan pengetahuan tentang pemrograman berorientasi objek (Object Oriented Programming).

Instalasi

MySQL, Apache2, PHP5, mcrypt

MySQL

```
sudo apt-get install mysql-server mysql-client
```

Setelah mysql terinstall kita dapat mengeceknya dengan mengetikkan petintah berikut di terminal

```
mysql -V

mysql  Ver 14.14 Distrib 5.5.43, for debian-linux-gnu (x86_64) using
readline 6.3
```

Kita sudah berhasil memasang mysql dengan versi 5.5

Apache2

```
sudo apt-get install apache2
```

Setelah apache terinstall kita dapat mengeceknya dengan mengetikkan perintah berikut di terminal

```
apache2 -v

Server version: Apache/2.4.7 (Ubuntu)
Server built:   Mar 10 2015 13:05:59
```

Kita sudah berhasil menginstall apache2 dengan versi 2.4.7, atau dapat kita cek pada browser dengan URL `http://localhost/` atau `http://127.0.0.1/`. Untuk direktori yang dipasang oleh apache2 ini terdapat di `/var/www/html` dan konfigurasinya ada di `/etc/apache2`.

PHP5

```
sudo apt-get install php5 libapache2-mod-php5
```

Setelah php5 terinstall kita dapat mengeceknya dengan perintah berikut di terminal

```
php5 -v

PHP 5.5.9-1ubuntu4.9 (cli) (built: Apr 17 2015 11:44:57)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
   with Zend OPcache v7.0.3, Copyright (c) 1999-2014, by Zend
Technologies
```

Kemudian kita akan restart apache2 agar dapat bekerja dengan PHP5

```
sudo service apache2 restart
```

Terdapat tambahan untuk php5 ini agar dapat berjalan dengan baik dengan ekstensi yang harus kita install dengan mengetikkan perintah berikut di terminal

```
sudo apt-get install php5-mysql php5-curl php5-gd php5-intl php-pear
php5-imagick php5-imap php5-mcrypt php5-memcache php5-ming php5-ps
php5-pspell php5-recode php5-snmp php5-sqlite php5-tidy php5-xmlrpc
php5-xsl
```

Kemudian kita restart lagi agar semua terintegrasi

```
sudo service apache2 restart
```

mcrypt

Agar dapat memastikan mcrypt sudah dapat digunakan kita gunakan perintah berikut di terminal

```
sudo php5enmod mcrypt
```

Composer

Unduh Composer

Kita dapat mengunduh composer untuk linux dengan mengetikkan perintah berikut pada terminal

```
sudo curl -sS https://getcomposer.org/installer | php
```

Atau

```
php -r "readfile('https://getcomposer.org/installer')" | php
```

Kemudian proses instalasi composer akan dimulai

```
#!/usr/bin/env php
All settings correct for using Composer
Downloading...

Composer successfully installed to: /home/<username>/composer.phar
Use it: php composer.phar
```

Kita sudah berhasil mengunduh composer.

Penggunaan secara global

Saat ini composer yang kita unduh berada di `/home/<username>/composer.phar`. Agar dapat digunakan secara global kita akan pindahkan ke folder `/usr/local/bin` dengan mengetikkan perintah berikut

```
sudo mv composer.phar /usr/local/bin/composer
```

Sehingga apabila kita inginmenggunakan composer kita cukup mengetikkan composer saja pada terminal.

Oke, kita sudah melakukan proses awal untuk instalasi Laravel, tapi tenang saja, proses instalasi ini hanya dilakukan sekali dan tidak dilakukan lagi untuk menginstalan Laravel berikutnya.

Laravel

Via Laravel Installer

Dengan laravel installer, kita akan cukup mudah untuk melakukan pembuatan aplikasi baru laravel. Untuk menginstall laravel installer, gunakan perintah berikut ini

```
composer global require "laravel/installer=~1.1"
```

Berikut ini adalah proses pengunduhan laravel installer

```
Changed current directory to /home/ariefsetya/.composer
./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing symfony/process (v2.7.2)
    Downloading: 100%

  - Installing symfony/console (v2.7.2)
```

```
    Downloading: 100%

  - Installing react/promise (v2.2.1)
    Downloading: 100%

  - Installing guzzlehttp/streams (3.0.0)
    Downloading: 100%

  - Installing guzzlehttp/ringphp (1.1.0)
    Downloading: 100%

  - Installing guzzlehttp/guzzle (5.3.0)
    Downloading: 100%

  - Installing laravel/installer (v1.2.1)
    Downloading: 100%

symfony/console suggests installing symfony/event-dispatcher ()
symfony/console suggests installing psr/log (For using the console
logger)
Writing lock file
Generating autoload files
```

Setelah itu kita tambahkan `/.composer/vendor/bin` ke PATH agar laravel dapat dieksekusi secara langsung dengan perintah berikut

```
PATH=$PATH:~/.composer/vendor/bin/
```

Selanjutnya kita arahkan direktori kita ke `/var/www/html`

```
cd /var/www/html
```

Kemudian kita buat aplikasi baru laravel

```
laravel new <nama_project>
```

Maka akan mulai mengunduh laravel seperti dibawah ini

```
Crafting application...
> php -r "copy('.env.example', '.env');"
> php artisan clear-compiled
> php artisan optimize
Generating optimized class loader
> php artisan key:generate
Application key [QevAuRjrbtJLBhAqO0E1XBHLP8I7cnut] set
```

```
successfully.
Application ready! Build something amazing
```

Kemudian kita buat permission agar dapat diakses

```
sudo chmod -R 755 /var/www/html/<nama_project>/
sudo chmod -R 777 /var/www/html/<nama_project>/storage/
```

Dengan menggunakan laravel installer, kita akan mendapatkan versi terakhir dari laravel, jadi apabila kita jalankan perintah berikut pada folder laravel

```
php artisan -V
```

Maka akan muncul versi terakhir, untuk saat ini adalah versi 5.1.7 (LTS)

```
Laravel Framework version 5.1.7 (LTS)
```

Via composer create-project

Cara kedua untuk mengunduh laravel adalah dengan cara berikut ini

```
Composer create-project laravel/laravel
/var/www/html/<nama_project>
```

Kemudian akan memulai pengunduhan laravel

```
Installing laravel/laravel (v5.0.0)
  - Installing laravel/laravel (v5.0.0)
    Loading from cache

Created project in /var/www/html/<nama_project>
Loading composer repositories with package information
Installing dependencies (including require-dev)
  - Installing vlucas/phpdotenv (v1.1.1)
    Loading from cache

  - Installing symfony/var-dumper (v2.6.10)
    Downloading: 100%
```

```
- Installing symfony/translation (v2.6.10)
  Downloading: 100%

- Installing symfony/security-core (v2.6.10)
  Downloading: 100%

- Installing symfony/routing (v2.6.10)
  Downloading: 100%

- Installing symfony/process (v2.6.10)
  Downloading: 100%

- Installing psr/log (1.0.0)
  Loading from cache

- Installing symfony/debug (v2.6.10)
  Downloading: 100%

- Installing symfony/http-foundation (v2.6.10)
  Downloading: 100%

- Installing symfony/event-dispatcher (v2.7.2)
  Downloading: 100%

- Installing symfony/http-kernel (v2.6.10)
  Downloading: 100%

- Installing symfony/finder (v2.6.10)
  Downloading: 100%

- Installing symfony/console (v2.6.10)
  Downloading: 100%

- Installing swiftmailer/swiftmailer (v5.4.1)
  Loading from cache

- Installing jakub-onderka/php-console-color (0.1)
  Loading from cache

- Installing jakub-onderka/php-console-highlighter (v0.3.2)
  Loading from cache

- Installing dnoegel/php-xdg-base-dir (0.1)
  Loading from cache

- Installing nikic/php-parser (v1.4.0)
  Downloading: 100%

- Installing psy/psysh (v0.4.4)
  Loading from cache

- Installing nesbot/carbon (1.20.0)
  Loading from cache

- Installing mtdowling/cron-expression (v1.0.4)
```

```
  Loading from cache

- Installing monolog/monolog (1.15.0)
  Downloading: 100%

- Installing league/flysystem (1.0.9)
  Downloading: 100%

- Installing jeremeamia/superclosure (2.1.0)
  Loading from cache

- Installing ircmaxell/password-compat (v1.0.4)
  Loading from cache

- Installing doctrine/inflector (v1.0.1)
  Loading from cache

- Installing danielstjules/stringy (1.9.0)
  Loading from cache

- Installing symfony/filesystem (v2.7.2)
  Downloading: 100%

- Installing classpreloader/classpreloader (1.4.0)
  Loading from cache

- Installing laravel/framework (v5.0.33)
  Loading from cache

- Installing sebastian/version (1.0.6)
  Loading from cache

- Installing sebastian/global-state (1.0.0)
  Loading from cache

- Installing sebastian/recursion-context (1.0.0)
  Loading from cache

- Installing sebastian/exporter (1.2.0)
  Loading from cache

- Installing sebastian/environment (1.2.2)
  Loading from cache

- Installing sebastian/diff (1.3.0)
  Loading from cache

- Installing sebastian/comparator (1.1.1)
  Loading from cache

- Installing symfony/yaml (v2.7.2)
  Downloading: 100%

- Installing doctrine/instantiator (1.0.5)
  Loading from cache
```

```
  - Installing phpdocumentor/reflection-docblock (2.0.4)
    Loading from cache

  - Installing phpspec/prophecy (v1.4.1)
    Loading from cache

  - Installing phpunit/php-text-template (1.2.1)
    Loading from cache

  - Installing phpunit/phpunit-mock-objects (2.3.5)
    Loading from cache

  - Installing phpunit/php-timer (1.0.6)
    Loading from cache

  - Installing phpunit/php-token-stream (1.4.3)
    Loading from cache

  - Installing phpunit/php-file-iterator (1.4.0)
    Loading from cache

  - Installing phpunit/php-code-coverage (2.1.8)
    Downloading: 100%

  - Installing phpunit/phpunit (4.7.7)
    Downloading: 100%

  - Installing phpspec/php-diff (v1.0.2)
    Loading from cache

  - Installing phpspec/phpspec (2.2.1)
    Loading from cache

symfony/var-dumper suggests installing ext-symfony_debug ()
symfony/translation suggests installing symfony/config ()
symfony/security-core suggests installing symfony/validator (For
using the user password constraint)
symfony/security-core suggests installing
symfony/expression-language (For using the expression voter)
symfony/routing suggests installing symfony/config (For using the
all-in-one router or any loader)
symfony/routing suggests installing symfony/expression-language
(For using expression matching)
symfony/routing suggests installing doctrine/annotations (For using
the annotation loader)
symfony/event-dispatcher suggests installing
symfony/dependency-injection ()
symfony/http-kernel suggests installing symfony/browser-kit ()
symfony/http-kernel suggests installing symfony/class-loader ()
symfony/http-kernel suggests installing symfony/config ()
symfony/http-kernel suggests installing
symfony/dependency-injection ()
psy/psysh suggests installing ext-pdo-sqlite (The doc command
requires SQLite to work.)
monolog/monolog suggests installing graylog2/gelf-php (Allow
sending log messages to a GrayLog2 server)
```

```
monolog/monolog suggests installing raven/raven (Allow sending log
messages to a Sentry server)
monolog/monolog suggests installing doctrine/couchdb (Allow sending
log messages to a CouchDB server)
monolog/monolog suggests installing ruflin/elastica (Allow sending
log messages to an Elastic Search server)
monolog/monolog suggests installing videlalvaro/php-amqplib (Allow
sending log messages to an AMQP server using php-amqplib)
monolog/monolog suggests installing ext-amqp (Allow sending log
messages to an AMQP server (1.0+ required))
monolog/monolog suggests installing ext-mongo (Allow sending log
messages to a MongoDB server)
monolog/monolog suggests installing aws/aws-sdk-php (Allow sending
log messages to AWS services like DynamoDB)
monolog/monolog suggests installing rollbar/rollbar (Allow sending
log messages to Rollbar)
monolog/monolog suggests installing php-console/php-console (Allow
sending log messages to Google Chrome)
league/flysystem suggests installing
league/flysystem-eventable-filesystem (Allows you to use
EventableFilesystem)
league/flysystem suggests installing league/flysystem-rackspace
(Allows you to use Rackspace Cloud Files)
league/flysystem suggests installing league/flysystem-copy (Allows
you to use Copy.com storage)
league/flysystem suggests installing league/flysystem-azure
(Allows you to use Windows Azure Blob storage)
league/flysystem suggests installing league/flysystem-webdav
(Allows you to use WebDAV storage)
league/flysystem suggests installing league/flysystem-aws-s3-v2
(Allows you to use S3 storage with AWS SDK v2)
league/flysystem suggests installing league/flysystem-aws-s3-v3
(Allows you to use S3 storage with AWS SDK v3)
league/flysystem suggests installing league/flysystem-dropbox
(Allows you to use Dropbox storage)
league/flysystem suggests installing
league/flysystem-cached-adapter (Flysystem adapter decorator for
metadata caching)
league/flysystem suggests installing league/flysystem-sftp (Allows
you to use SFTP server storage via phpseclib)
league/flysystem suggests installing league/flysystem-ziparchive
(Allows you to use ZipArchive adapter)
laravel/framework suggests installing aws/aws-sdk-php (Required to
use the SQS queue driver and SES mail driver (~2.4).)
laravel/framework suggests installing doctrine/dbal (Required to
rename columns and drop SQLite columns (~2.4).)
laravel/framework suggests installing guzzlehttp/guzzle (Required
to use the Mailgun and Mandrill mail drivers (~5.0).)
laravel/framework suggests installing iron-io/iron_mq (Required to
use the iron queue driver (~1.5).)
laravel/framework suggests installing league/flysystem-aws-s3-v2
(Required to use the Flysystem S3 driver (~1.0).)
laravel/framework suggests installing league/flysystem-rackspace
(Required to use the Flysystem Rackspace driver (~1.0).)
laravel/framework suggests installing pda/pheanstalk (Required to
use the beanstalk queue driver (~3.0).)
```

```
laravel/framework suggests installing predis/predis (Required to
use the redis cache and queue drivers (~1.0).)
sebastian/global-state suggests installing ext-uopz (*)
phpdocumentor/reflection-docblock suggests installing
dflydev/markdown (~1.0)
phpdocumentor/reflection-docblock suggests installing
erusev/parsedown (~1.0)
phpunit/php-code-coverage suggests installing ext-xdebug (>=2.2.1)
phpunit/phpunit suggests installing phpunit/php-invoker (~1.1)
phpspec/phpspec suggests installing phpspec/nyan-formatters (~1.0
- Adds Nyan formatters)
Writing lock file
Generating autoload files
> php artisan clear-compiled
> php artisan optimize
Generating optimized class loader
Compiling common classes
> php -r "copy('.env.example', '.env');"
> php artisan key:generate
Application key [UjoBrJ0XcbJHJ5VliJTkL7LD2e3TQc6J] set
successfully.
```

Kemudian kita buat permission agar dapat diakses

```
sudo chmod -R 755 /var/www/html/<nama_project>/
sudo chmod -R 777 /var/www/html/<nama_project>/storage/
```

Dengan menggunakan laravel installer, kita akan mendapatkan versi terakhir dari laravel, jadi apabila kita jalankan perintah berikut pada folder laravel

```
php artisan -V
```

Maka akan muncul versi terakhir, untuk saat ini adalah versi 5.0.33

```
Laravel Framework version 5.0.33
```

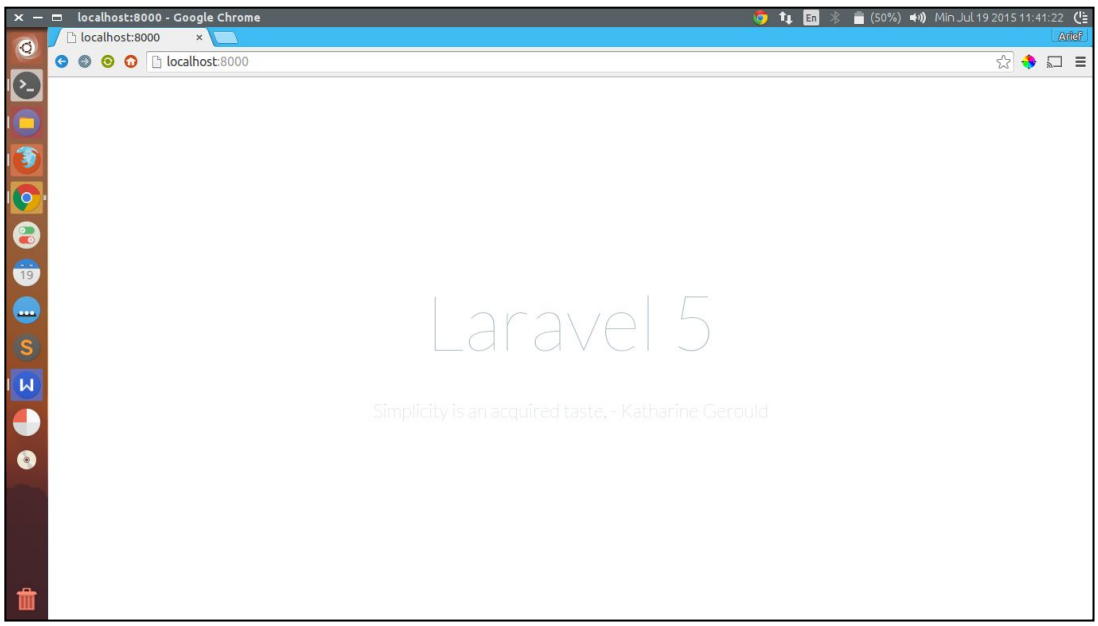
Oke, kita sudah selesai pada proses instalasi laravel dan kita bisa mencoba menjalankan aplikasi laravel secara langsung dengan menggunakan perintah berikut ini

```
php artisan serve
```

Maka yang kita dapatkan adalah berikut ini

```
Laravel development server started on http://localhost:8000/
```

Sehingga kita bisa akses aplikasi laravel kita pada browser menggunakan URL diatas



Konfigurasi

Virtual Server

Ada beberapa cara untuk melakukan konfigurasi virtual server untuk laravel, untuk yang akan kita bahas kali ini ada 2, yaitu Homestead dan Apache2. Yuk simak...

Homestead

Menggunakan Homestead, sebelum menggunakan tools yang free dan open source ini, kita harus menginstall VirtualBox dan Vagrant. Akan tetapi resource yang dibutuhkan sangat besar hingga mencapai 1GB, maka dari itu kita akan lebih mudah menggunakan Apache2. Untuk VirtualBox kita dapat mengunduhnya di link berikut ini

```
https://www.virtualbox.org/wiki/Downloads
```

Serta vagrant yang bisa kita unduh pada link berikut ini

```
http://www.vagrantup.com/downloads.html
```

Setelah VirtualBox dan vagrant terinstall kita akan mulai dengan perintah berikut untuk menambahkan vagrant box

```
vagrant box add laravel/homestead
```

Maka proses download akan berjalan, apabila menemui pilihan VirtualBox atau VMWare kita bisa pilih VirtualBox.

```
Vagrant is upgrading some internal state for the latest version.
Please do not quit Vagrant at this time. While upgrading, Vagrant
will need to copy all your boxes, so it will use a considerable
```

```
amount of disk space. After it is done upgrading, the temporary disk
space will be freed.

Press ctrl-c now to exit if you want to remove some boxes or free
up some disk space.

Press the Enter or Return key to continue.
==> box: Loading metadata for box 'laravel/homestead'
   box: URL: https://atlas.hashicorp.com/laravel/homestead
This box can work with multiple providers! The providers that it
can work with are listed below. Please review the list and choose
the provider you will be working with.

1) virtualbox
2) vmware_desktop

Enter your choice: 1
==> box: Adding box 'laravel/homestead' (v0.2.7) for provider:
virtualbox
   box: Downloading:
https://atlas.hashicorp.com/laravel/boxes/homestead/versions/0.2.
7/providers/virtualbox.box
   box: Progress: 0% (Rate: 4299k/s, Estimated time remaining:
0:03:56)
```

Setelah selesai, kita clone laravel/homestead

```
git clone https://github.com/laravel/homestead.git Homestead
```

Kemudian jalankan perintah berikut dari direktori Homestead untuk membuat file konfigurasi homestead.yaml

```
bash init.sh
```

Setelah itu kita bisa lakukan konfigurasi virtual server kita dengan mengedit file homestead.yaml. Selanjutnya adalah kita akan atur SSH key untuk virtual server kita

```
ssh-keygen -t rsa -C "you@homestead"
```

Setelah itu kita tambahkan hosts agar dapat diakses dengan mudah, contohnya seperti dibawah ini, file hosts pada linux ada di `/etc/hosts`

```
<ip_addr>  <hostname>
```

Sehingga kita sudah dapat mengakses aplikasi kita melalui browser menggunakan URL berikut

```
http://<hostname>
```

Apache2

Untuk membuat virtual server menggunakan apache2, kita tidak perlu mengunduh file atau aplikasi lain yang membutuhkan resource yang sangat besar. Oke langsung saja, pertama kita tambahkan virtual server pada direktori `sites-available` apache

```
sudo gedit /etc/apache2/sites-available/<hostname>.conf
```

Kemudian kita tuliskan script berikut ini

```
<VirtualHost *:<port>>

 ServerName <hostname>

 ServerAdmin <username>@<hostname>
 DocumentRoot /var/www/html/<project_name>/public
 Options FollowSymLinks Indexes Multiviews

        <Directory /var/www/html/<project_name>/public>
               AllowOverride All
               Order allow,deny
               Allow from all
        </Directory>

 ErrorLog ${APACHE_LOG_DIR}/error.log
 CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Selanjutnya kita aktifkan dengan perintah berikut ini

```
sudo a2ensite <hostname>.conf
```

Kita akan mendapatkan pesan seperti berikut ini

```
Enabling site <hostname>.
To activate the new configuration, you need to run:
  service apache2 reload
```

Kemudian kita reload service apache2 kita agar virtual server dapat terintegrasi

```
sudo service apache2 reload
 * Reloading web server apache2
 *
```

Setelah itu kita tambahkan hostname kita ke file hosts yang ada di `/etc/hosts`

```
<ip> <hostname>
```

Nah, kita sudah selesai, kita dapat mengakses aplikasi kita melalui

```
http://<hostname>
```

Tutorial

Basic

Artisan

Di Laravel, kita mendapatkan sebuah program command-line bernama artisan, disini kita dapat membuat file-file untuk controller, model dan yang lainnya dengan mudah, untuk menggunakannya kita dapat menggunakan perintah berikut ketika berada di dalam folder aplikasi

```
php artisan command [options] [arguments]
```

Apabila kita mengetikkan perintah php artisan, maka akan muncul help dari penggunaan artisan tersebut, atau seperti yang tampak dibawah ini

```
Laravel Framework version 5.0.33

Usage:
 command [options] [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 output, 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
 --env                 The environment the command should run under.

Available commands:
 clear-compiled        Remove the compiled class file
 down                  Put the application into maintenance mode
 env                   Display the current framework environment
 fresh                 Remove the scaffolding included with the
framework
 help                  Displays help for a command
 inspire               Display an inspiring quote
 list                  Lists commands
 migrate               Run the database migrations
 optimize              Optimize the framework for better performance
 serve                 Serve the application on the PHP development
```

```
server
 tinker                Interact with your application
 up                    Bring the application out of maintenance mode
app
 app:name              Set the application namespace
auth
 auth:clear-resets     Flush expired password reset tokens
cache
 cache:clear           Flush the application cache
 cache:table           Create a migration for the cache database
table
config
 config:cache          Create a cache file for faster configuration
loading
 config:clear          Remove the configuration cache file
db
 db:seed               Seed the database with records
event
 event:generate        Generate the missing events and handlers
based on registration
handler
 handler:command       Create a new command handler class
 handler:event         Create a new event handler class
key
 key:generate          Set the application key
make
 make:command          Create a new command class
 make:console          Create a new Artisan command
 make:controller       Create a new resource controller class
 make:event            Create a new event class
 make:middleware       Create a new middleware class
 make:migration        Create a new migration file
 make:model            Create a new Eloquent model class
 make:provider         Create a new service provider class
 make:request          Create a new form request class
migrate
 migrate:install       Create the migration repository
 migrate:refresh       Reset and re-run all migrations
 migrate:reset         Rollback all database migrations
 migrate:rollback      Rollback the last database migration
 migrate:status        Show the status of each migration
queue
 queue:failed          List all of the failed queue jobs
 queue:failed-table    Create a migration for the failed queue jobs
database table
 queue:flush           Flush all of the failed queue jobs
 queue:forget          Delete a failed queue job
 queue:listen          Listen to a given queue
 queue:restart         Restart queue worker daemons after their
current job
 queue:retry           Retry a failed queue job
 queue:subscribe       Subscribe a URL to an Iron.io push queue
 queue:table           Create a migration for the queue jobs database
table
 queue:work            Process the next job on a queue
route
 route:cache           Create a route cache file for faster route
```

```
registration
 route:clear            Remove the route cache file
 route:list             List all registered routes
schedule
 schedule:run           Run the scheduled commands
session
 session:table          Create a migration for the session database
table
vendor
 vendor:publish         Publish any publishable assets from vendor
packages
```

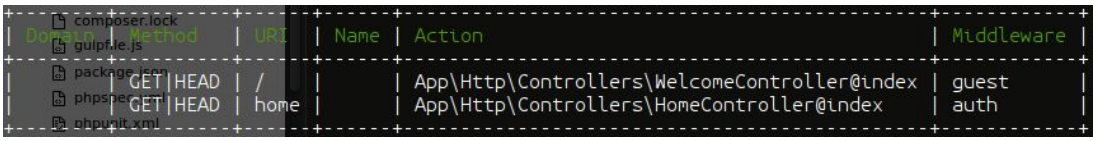
Misalnya kita ingin membuat controller dengan nama Siswa, maka kita cukup menggunakan perintah dibawah ini

```
php artisan make:controller SiswaController
Controller created successfully.
```

Atau melihat route yang ada

```
php artisan route:list
```

Hasilnya



Enviroment

Di laravel, kita akan mendapatkan file dot env (.env) di direktori utama laravel, disini kita dapat mengatur hal-hal yang dibutuhkan agar tidak terlalu banyak konfigurasi, misalnya untuk konfigurasi database, ada host, username, password dan nama database, kemudian untuk pengaturan apakah aplikasi masih di lokal atau sudah rilis di hosting atau konfigurasi agar dapat menampilkan error atau tidak. Berikut ini isi dari file .env

```
APP_ENV=local
APP_DEBUG=true
```

```
APP_KEY=UjoBrJ0XcbJHJ5VliJTkL7LD2e3TQc6J

DB_HOST=localhost
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

CACHE_DRIVER=file
SESSION_DRIVER=file
```

Config App

Untuk konfigurasi aplikasi sendiri sebetulnya sudah hampir terwakili pada file .env, akan tetapi ada beberapa hal yang harus kita perhatikan. Yuk simak.

1. App.php

   Pada file ini kita dapat mengatur hal-hal dasar seperti debug aplikasi, url, zona waktu, lokalisasi bahasa, key aplikasi, cipher, log, provider yang digunakan dalam aplikasi, serta penggunaan alias untuk nama class agar tidak terlalu panjang dalam menuliskan nama class

```
<?php
return [
 'debug' => env('APP_DEBUG'),
 'url' => 'http://localhost',
 'timezone' => 'UTC',
 'locale' => 'en',
 'fallback_locale' => 'en',
 'key' => env('APP_KEY', 'SomeRandomString'),
 'cipher' => MCRYPT_RIJNDAEL_128,
 'log' => 'daily',
 'providers' => [
'Illuminate\Foundation\Providers\ArtisanServiceProvider',
'Illuminate\Auth\AuthServiceProvider',
'Illuminate\Bus\BusServiceProvider',
'Illuminate\Cache\CacheServiceProvider',
'Illuminate\Foundation\Providers\ConsoleSupportServiceProvider',
'Illuminate\Routing\ControllerServiceProvider',
'Illuminate\Cookie\CookieServiceProvider',
'Illuminate\Database\DatabaseServiceProvider',
'Illuminate\Encryption\EncryptionServiceProvider',
'Illuminate\Filesystem\FilesystemServiceProvider',
'Illuminate\Foundation\Providers\FoundationServiceProvider',
'Illuminate\Hashing\HashServiceProvider',
'Illuminate\Mail\MailServiceProvider',
'Illuminate\Pagination\PaginationServiceProvider',
```

```
'Illuminate\Pipeline\PipelineServiceProvider',
'Illuminate\Queue\QueueServiceProvider',
'Illuminate\Redis\RedisServiceProvider',
'Illuminate\Auth\Passwords\PasswordResetServiceProvider',
'Illuminate\Session\SessionServiceProvider',
'Illuminate\Translation\TranslationServiceProvider',
'Illuminate\Validation\ValidationServiceProvider',
'Illuminate\View\ViewServiceProvider',
'App\Providers\AppServiceProvider',
'App\Providers\BusServiceProvider',
'App\Providers\ConfigServiceProvider',
'App\Providers\EventServiceProvider',
'App\Providers\RouteServiceProvider',
 ],
 'aliases' => [
        'App'      => 'Illuminate\Support\Facades\App',
        'Artisan'   => 'Illuminate\Support\Facades\Artisan',
        'Auth'     => 'Illuminate\Support\Facades\Auth',
        'Blade'    => 'Illuminate\Support\Facades\Blade',
        'Bus'      => 'Illuminate\Support\Facades\Bus',
        'Cache'    => 'Illuminate\Support\Facades\Cache',
        'Config'   => 'Illuminate\Support\Facades\Config',
        'Cookie'   => 'Illuminate\Support\Facades\Cookie',
        'Crypt'    => 'Illuminate\Support\Facades\Crypt',
        'DB'       => 'Illuminate\Support\Facades\DB',
        'Eloquent'  => 'Illuminate\Database\Eloquent\Model',
        'Event'    => 'Illuminate\Support\Facades\Event',
        'File'     => 'Illuminate\Support\Facades\File',
        'Hash'     => 'Illuminate\Support\Facades\Hash',
        'Input'    => 'Illuminate\Support\Facades\Input',
        'Inspiring' => 'Illuminate\Foundation\Inspiring',
        'Lang'     => 'Illuminate\Support\Facades\Lang',
        'Log'      => 'Illuminate\Support\Facades\Log',
        'Mail'     => 'Illuminate\Support\Facades\Mail',
        'Password'  => 'Illuminate\Support\Facades\Password',
        'Queue'    => 'Illuminate\Support\Facades\Queue',
        'Redirect'  => 'Illuminate\Support\Facades\Redirect',
        'Redis'    => 'Illuminate\Support\Facades\Redis',
        'Request'   => 'Illuminate\Support\Facades\Request',
        'Response'  => 'Illuminate\Support\Facades\Response',
        'Route'    => 'Illuminate\Support\Facades\Route',
        'Schema'   => 'Illuminate\Support\Facades\Schema',
        'Session'   => 'Illuminate\Support\Facades\Session',
        'Storage'   => 'Illuminate\Support\Facades\Storage',
        'URL'      => 'Illuminate\Support\Facades\URL',
        'Validator' => 'Illuminate\Support\Facades\Validator',
        'View'     => 'Illuminate\Support\Facades\View',
 ],
];
```

2. auth.php

Di laravel, kita sudah dapat menggunakan fitur autentikasi yang dapat digunakan untuk login register pengguna, apabila tidak

digunakan tidak akan mempengaruhi file-file lainnya. Berikut ini isi dari auth.php. Kita bisa mengatur driver database yang sedang digunakan, model, table dan pengaturan lain yang digunakan untuk autentikasi

```
<?php
return [
 'driver' => 'eloquent',
 'model' => 'App\User',
 'table' => 'users',
 'password' => [
        'email' => 'emails.password',
        'table' => 'password_resets',
        'expire' => 60,
 ],
];
```

3. cache.php

Di file ini kita bisa mengatur pengaturan untuk caching, seperti yang kita atur pada .env kita sudah mengatur cache agar menggunakan driver file. Berikut ini adalah isi dari file tersebut

```
<?php
return [
 'default' => env('CACHE_DRIVER', 'file'),
 'stores' => [
        'apc' => [
               'driver' => 'apc'
        ],
        'array' => [
               'driver' => 'array'
        ],
        'database' => [
               'driver' => 'database',
               'table'  => 'cache',
               'connection' => null,
        ],
        'file' => [
               'driver' => 'file',
               'path'   => storage_path().'/framework/cache',
        ],
        'memcached' => [
               'driver'  => 'memcached',
               'servers' => [
                      [
                             'host' => '127.0.0.1', 'port' => 11211,
'weight' => 100
                      ],
```

```
                ],
        ],
        'redis' => [
                'driver' => 'redis',
                'connection' => 'default',
        ],
 ],
 'prefix' => 'laravel',
];
```

4. compile.php

Ini adalah file yang akan selalu meload setiap class yang sudah didefinisikan didalamnya pada setiap request aplikasi, dengan menggunakan perintah `php artisan optimize`

```
<?php
return [
 'files' => [
realpath(__DIR__.'/../app/Providers/AppServiceProvider.php'),
realpath(__DIR__.'/../app/Providers/BusServiceProvider.php'),
realpath(__DIR__.'/../app/Providers/ConfigServiceProvider.php'),
realpath(__DIR__.'/../app/Providers/EventServiceProvider.php'),
realpath(__DIR__.'/../app/Providers/RouteServiceProvider.php'),
 ],
 'providers' => [
        //
 ],
];
```

5. database.php

Pada file ini, kita dapat mengatur koneksi database yang digunakan. Berikut ini adalah isi dari file tersebut

```
<?php
return [
 'fetch' => PDO::FETCH_CLASS,
 'default' => 'mysql',
 'connections' => [
        'sqlite' => [
                'driver'   => 'sqlite',
                'database' => storage_path().'/database.sqlite',
                'prefix'   => '',
        ],
        'mysql' => [
                'driver'    => 'mysql',
                'host'     => env('DB_HOST', 'localhost'),
                'database'  => env('DB_DATABASE', 'forge'),
                'username'  => env('DB_USERNAME', 'forge'),
```

```
            'password'  => env('DB_PASSWORD', ''),
            'charset'   => 'utf8',
            'collation' => 'utf8_unicode_ci',
            'prefix'    => '',
            'strict'    => false,
        ],
        'pgsql' => [
            'driver'   => 'pgsql',
            'host'     => env('DB_HOST', 'localhost'),
            'database' => env('DB_DATABASE', 'forge'),
            'username' => env('DB_USERNAME', 'forge'),
            'password' => env('DB_PASSWORD', ''),
            'charset'  => 'utf8',
            'prefix'   => '',
            'schema'   => 'public',
        ],
        'sqlsrv' => [
            'driver'   => 'sqlsrv',
            'host'     => env('DB_HOST', 'localhost'),
            'database' => env('DB_DATABASE', 'forge'),
            'username' => env('DB_USERNAME', 'forge'),
            'password' => env('DB_PASSWORD', ''),
            'prefix'   => '',
        ],
 ],
 'migrations' => 'migrations',
 'redis' => [
        'cluster' => false,
        'default' => [
            'host'     => '127.0.0.1',
            'port'     => 6379,
            'database' => 0,
        ],
 ],
];
```

6. filesystem.php

Konfigurasi ini digunakan untuk mendefinisikan jenis sistem file yang akan digunakan oleh aplikasi, berikut ini isi dari file tersebut

```
<?php
return [
 'default' => 'local',
 'cloud' => 's3',
 'disks' => [
        'local' => [
            'driver' => 'local',
            'root'   => storage_path().'/app',
        ],
        's3' => [
            'driver' => 's3',
```

```
                'key'    => 'your-key',
                'secret' => 'your-secret',
                'region' => 'your-region',
                'bucket' => 'your-bucket',
        ],
        'rackspace' => [
                'driver'   => 'rackspace',
                'username' => 'your-username',
                'key'      => 'your-key',
                'container' => 'your-container',
                'endpoint'  =>
'https://identity.api.rackspacecloud.com/v2.0/',
                'region'   => 'IAD',
        ],
 ],
];
```

7. mail.php

File ini digunakan untuk konfigurasi pengiriman email. Berikut ini adalah isi dari file tersebut

```
<?php
return [
 'driver' => 'smtp',
 'host' => 'smtp.mailgun.org',
 'port' => 587,
 'from' => ['address' => null, 'name' => null],
 'encryption' => 'tls',
 'username' => null,
 'password' => null,
 'sendmail' => '/usr/sbin/sendmail -bs',
 'pretend' => false,
];
```

8. services.php

File ini digunakan untuk konfigurasi service lain yang ingin digunakan pada aplikasi misalnya mailgun atau mandrill. Berikut isi dari file tersebut

```
<?php
return [
 'mailgun' => [
        'domain' => '',
        'secret' => '',
 ],
 'mandrill' => [
        'secret' => '',
```

```
 ],
 'ses' => [
        'key' => '',
        'secret' => '',
        'region' => 'us-east-1',
 ],
 'stripe' => [
        'model'  => 'User',
        'secret' => '',
 ],
];
```

9. view.php

File ini digunakan untuk mendefinisikan dimana kita akan menempatkan file template blade atau file views html kita. Berikut ini adalah isi dari file tersebut

```
<?php
return [
 'paths' => [
        realpath(base_path('resources/views'))
 ],
 'compiled' => realpath(storage_path().'/framework/views'),
];
```

File dan Folder utama

Disini kita akan mengenal file dan folder utama yang biasa digunakan untuk aplikasi sederhana

1. routes.php

   File ini berada pada folder `/app/Http/routes.php`. File ini digunakan untuk mendefinisikan seluruh rute yang akan digunakan oleh aplikasi menuju controller atau langsung menampilkan sesuatu.
2. app/Http/Controllers

   Folder ini berisi seluruh file dan folder controller yang digunakan untuk aplikasi.
3. app/Http/Middleware

   Folder ini berisi seluruh file dan folder yang digunakan untuk keperluan autentikasi yang digunakan pada aplikasi
4. app/Http/Requests

   Folder ini digunakan untuk menampung seluruh file yang digunakan untuk meneriman Form Request yang digunakan oleh aplikasi
5. app/<nama_model>.php

   Inilah yang sedikit membingungkan pada Laravel 5, dikarenakan tidak adanya folder model pada versi ini maka folder app digunakan sebagai tempat file-file model yang berhubungan dengan database.
6. config

   Ini adalah folder utama untuk melakukan konfigurasi pada aplikasi
7. database/migrations

   Folder ini berisi file-file untuk melakukan migration pada database.
8. public

   Pada folder ini halaman utama laravel diakses, yaitu berupa file index.php yang biasa digunakan untuk halaman utama sebuah

website

9. resources/views

   Pada folder ini seluruh file html atau template blade disimpan. Nantinya file ini akan dipanggil melalui controller.

10. storage

    Ini adalah tempat untuk menyimpan file-file hasil upload atau hasil generate dari sebuah tempate blade.

11. vendor

    Ini adalah folder yang memuat seluruh vendor-vendor yang digunakan oleh aplikasi.

Routing, Controller and Views

Sebelumnya kita akan mengenal terlebih dahulu struktur dari route itu sendiri, berikut penjelasannya

```
Route::get('/', 'WelcomeController@index');
```

- Route::get adalah sebuah fungsi untuk diteruskan ke file controller dengan sistem get
- '/' adalah URI yang kita dapat definisikan untuk diakses di browser
- 'WelcomeController' adalah file controller yang akan digunakan
- '@' adalah pemisah antara controller dan method
- 'index' adalah sebuah method atau fungsi yang akan dipanggil

Atau

```
Route::get('sekolah/{nama}', 'SekolahController@show');
```

- {nama} adalah variabel yang dapat diambil di controller dengan $nama

Atau juga dapat di definisikan seperti berikut

```
Route::get('sekolah/{nama}', function ($nama){
 echo $nama;
});
```

Route ini tidak harus menggunakan prefix Route::, apabila kita langsung menggunakan get saja atau menggunakan variabel seperti `$routes->get()` juga bisa.

Berikut ini tambahan penjelasan untuk contoh-contoh routes lainnya yang bisa digunakan.

Route GET sederhana

```
Route::get('/', function(){
    return 'Hello World';
});
```

Route sederhana lainnya

```
Route::post('foo/bar', function(){
    return 'Hello World';
});

Route::put('foo/bar', function(){
    //
});

Route::delete('foo/bar', function(){
    //
});
```

Membuat Route diterima oleh beberapa method

```
Route::match(['get', 'post'], '/', function(){
    return 'Hello World';
});
```

Membuat Route diterima pada semua request

```
Route::any('foo', function(){
    return 'Hello World';
});
```

Route dengan parameter

```
Route::get('user/{id}', function($id){
    return 'User '.$id;
});
```

Route dengan parameter default kosong

```
Route::get('user/{name?}', function($name = null){
    return $name;
});
```

Route dengan parameter default

```
Route::get('user/{name?}', function($name = 'John'){
    return $name;
});
```

Route dengan parameter yang sudah dalam RegEx

```
Route::get('user/{name}', function($name){
    //
})->where('name', '[A-Za-z]+');

Route::get('user/{id}', function($id){
    //
})->where('id', '[0-9]+');
```

Membuat RegEx untuk parameter dalam array

```
Route::get('user/{id}/{name}', function($id, $name){
    //
})->where(['id' => '[0-9]+', 'name' => '[a-z]+']);
```

Mendefinisikan RegEx Parameter secara default

```
$router->pattern('id', '[0-9]+');
```

Ketika sudah di definisikan, maka semua bentuk variabel akan berpengaruh

```
Route::get('user/{id}', function($id){
    // Only called if {id} is numeric.
});
```

Menamakan Route

```
Route::get('user/profile', ['as' => 'profile', function(){
    //
}]);
```

Kita juga dapat memberi nama yang spesifik pada sebuah method di controller

```
Route::get('user/profile', [
    'as' => 'profile', 'uses' => 'UserController@showProfile']
);
```

Sekarang kita bisa menggunakan nama route untuk redirect

```
$url = route('profile');
$redirect = redirect()->route('profile');
```

Untuk melihat nama route yang sedang menangani request, kita dapat menggunakan script berikut

```
$name = Route::currentRouteName();
```

Mengelompokkan Route untuk Middleware

```
Route::group(['middleware' => ['foo', 'bar']], function(){
Route::get('/', function()
{
        // Has Foo And Bar Middleware
});

Route::get('user/profile', function()
{
        // Has Foo And Bar Middleware
});
});
```

Mengelompokkan Route berdasarkan namespace

```
Route::group(['namespace' => 'Admin'], function(){
// Controllers Within The "App\Http\Controllers\Admin"
// Namespace

        Route::group(['namespace' => 'User'], function()
        {
        // Controllers Within The "App\Http\Controllers\Admin\User"
        // Namespace
        });

});
```

Membuat Route untuk sub-domain

```
Route::group(['domain' => '{account}.myapp.com'], function(){

        Route::get('user/{id}', function($account, $id)
        {
                //
        });
});
```

Prefix Route

```
Route::group(['prefix' => 'admin'], function(){
        Route::get('users', function()
        {
                // Matches The "/admin/users" URL
        });
});
```

Membuat parameter di dalam prefix Route

```
Route::group(['prefix' => 'accounts/{account_id}'], function(){
        Route::get('detail', function($account_id)
        {
                //
        });
});
```

Kita juga dapat membuat fungsi RegEx untuk prefix Route

```
Route::group([
    'prefix' => 'accounts/{account_id}',
    'where' => ['account_id' => '[0-9]+'],], function() {

    // Define Routes Here
});
```

Membuat Route Model Binding

Inilah kemudahan menggunakan Laravel, kita tidak perlu membuat fungsi utama untuk read data dari database, kita cukup menggunakan route model binding.

```
public function boot(Router $router){
    parent::boot($router);

    $router->model('user', 'App\User');
}
```

Selanjutnya, kita dapat menggunakan variabel user untuk hal lainnya

```
Route::get('profile/{user}', function(App\User $user){
    //
});
```

Kali ini kita akan membuat routing sederhana yang akan tersambung dengan controller dan views. Pertama kita akan buat route ke /sekolah dengan menambahkan script berikut ke file routes.php sehingga kita mengakses `http://<hostname>/sekolah`

```
Route::get('sekolah',function ()
{
 echo "ini halaman sekolah";
});
```

Sehingga apabila kita akses URLnya akan tampil seperti dibawah ini

Sekarang kita ingin menambahkan variabel agar URL dapat mengambil data dari URL dengan manambahkan script berikut ini

```
Route::get('sekolah/{nama}',function ($nama)
{
 echo "ini sekolah ".$nama;
});
```

Sehingga apabila kita akses URLnya akan tampil seperti dibawah ini

Sekarang kita akan coba buat sebuah controller bernama SekolahController.php dengan perintah berikut melalui terminal

```
php artisan make:controller SekolahController
```

Kita dapat cek pada folder controller bahwa file nya sudah terbuat dan berisi seperti ini

```
<?php namespace App\Http\Controllers;

use App\Http\Requests;
use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

class SekolahController extends Controller {

 /**
  * Display a listing of the resource.
  *
  * @return Response
  */
 public function index()
 {
        //
 }

 /**
  * Show the form for creating a new resource.
  *
  * @return Response
  */
 public function create()
 {
        //
 }

 /**
  * Store a newly created resource in storage.
  * @return Response
  */
 public function store()
 {
        //
 }

 /**
  * Display the specified resource.
  * @param  int  $id
  * @return Response
  */
 public function show($id)
 {
        //
 }

 /**
  * Show the form for editing the specified resource.
  * @param  int  $id
  * @return Response
  */
```

```
 public function edit($id)
 {
        //
 }

 /**
  * Update the specified resource in storage.
  * @param  int  $id
  * @return Response
  */
 public function update($id)
 {
        //
 }

 /**
  * Remove the specified resource from storage.
  *
  * @param  int  $id
  * @return Response
  */
 public function destroy($id)
 {
        //
 }

}
```

Namun apabila kita ingin membuat class tanpa isi yang terlalu banyak kita dapat menggunakan perintah dibawah ini

```
php artisan make:controller SekolahController --plain
```

Sehingga kita akan dapatkan controller seperti dibawah ini

```
<?php namespace App\Http\Controllers;

use App\Http\Requests;
use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

class SekolahController extends Controller {

 //

}
```

Nah sekarang kita akan langsung menggunakan SekolahController ini langsung dari Route, caranya adalah dengan mengubah sedikit controller yang kita buat sebelumnya menjadi seperti dibawah ini

```
Route::get('sekolah', 'SekolahController@index');
Route::get('sekolah/{nama}', 'SekolahController@show');
```

Pada script diatas kita akan memanggil fungsi atau method index yang ada di dalam class SekolahController. Kemudian kita tambahkan script echo pada method index di file SekolahController

```
public function index()
{
        echo "Ini halaman index sekolah";
}
```

Kemudian kita coba akses di browsernya dengan URL http://<hostname>/sekolah maka akan seperti gambar dibawah ini

Selanjutnya kita tambahkan script echo pada method show, jangan lupa ubah parameternya dan disamakan dengan variabel yang ada di route

```
public function show($nama)
{
        echo "Ini sekolah ".$nama;
}
```

Jika kita buka pada browser maka akan muncul seperti dibawah ini

Nah selanjutnya dari controller kita akan sambungkan ke view. Buat file baru pada folder `resources/views` dengan nama sekolah.blade.php. Nah inilah yang disebut template blade, kita harus menamakan file view dengan tambahan .blade dibelakangnya, tapi ketika kita panggil filenya kita cukup panggil dengan nama file utamanya saja. Misalnya nama filenya `sekolah.blade.php` maka kita akan panggil pada controller dengan `sekolah` saja.

Pada file sekolah.blade.php kita bisa isikan dengan tulisan biasa dahulu seperti

```
<!DOCTYPE html>
<html>
	<head>
		<title>Halaman sekolah</title>
	</head>
	<body>
		ini halaman view blade sekolah
	</body>
</html>
```

Kemudian kita ubah sedikit pada file controller dibagian method index dengan script berikut

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

Sehingga apabila berhasil maka akan muncul seperti dibawah ini

Tambahan :

Apabila file controllers atau views yang kita ingin buka ada di dalam folder, kita bisa menggunakan 2 cara yaitu menggunakan slash (/) seperti biasa dan menggunakan titik (.). Contohnya

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

Nah kita sudah tuntas membahas mengenai route, controller dan views. Untuk selanjutnya kita akan membahas mengenai sistem template blade.

Sistem template blade

Mengenal lebih jauh mengenai sistem ini, laravel menggunakan template blade sejak laravel 2. Dari template yang kita buat pada folder views nantinya akan di generate menjadi halaman php statis yang disimpan pada folder `storage/framework/views`, di folder ini kita akan mendapatkan file dengan nama yang sudah di encrypt. Pada template ini, ada beberapa fungsi php yang sudah dikemas rapi untuk digunakan. Yuk simak.

Mendefiniskan sebuah layout utama blade

```
<!-- Stored in resources/views/layouts/master.blade.php -->
<html>
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        @section('sidebar')
            This is the master sidebar.
        @show

        <div class="container">
            @yield('content')
        </div>
    </body></html>
```

Menggunakan layout utama blade

```
@extends('layouts.master')

@section('title', 'Page Title')

@section('sidebar')
    @parent

    <p>This is appended to the master sidebar.</p>
@stop

@section('content')
    <p>This is my body content.</p>
@stop
```

Apabila di dalam yield tidak ada konten yang diambil, kita dapat mendefinisikan secara default yang akan di munculkan oleh yield dengan script berikut

```
@yield('section', 'Default Content')
```

Echo Data

```
Hello, {{ $name }}.

The current UNIX timestamp is {{ time() }}.
```

Echo Data dengan eksistensi variabel

Biasanya kita dapat mengecek ada tidaknya sebuah variabel dengan script berikut

```
{{ isset($name) ? $name : 'Default' }}
```

Akan tetapi pada template blade, kita dapat menggunakan shortcut berikut

```
{{ $name or 'Default' }}
```

Menampilkan teks di dalam 2 kurung kurawal

```
@{{ This will not be processed by Blade }}
```

Menampilkan data tanpa fungsi escape string

```
Hello, {!! $name !!}.
```

Penggunaan if

```
@if (count($records) === 1)
    I have one record!
@elseif (count($records) > 1)
    I have multiple records!
@else
    I don't have any records!
@endif

@unless (Auth::check())
    You are not signed in.
@endunless
```

Perulangan

```
@for ($i = 0; $i < 10; $i++)
    The current value is {{ $i }}
@endfor

@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach

@forelse($users as $user)
    <li>{{ $user->name }}</li>
@empty
    <p>No users</p>
@endforelse

@while (true)
    <p>I'm looping forever.</p>
@endwhile
```

Include Sub Views

```
@include('view.name')
You may also pass an array of data to the included view:
@include('view.name', ['some' => 'data'])
```

Overwrite Section

```
@extends('list.item.container')

@section('list.item.content')
   <p>This is an item of type {{ $item->type }}</p>
@overwrite
```

Menampilkan bahasa

```
@lang('language.line')

@choice('language.line', 1)
```

Comments

```
{{-- This comment will not be in the rendered HTML --}}
```

Passing data ke Views

Ada beberapa cara untuk meneruskan data dari controller atau dari route ke views. Yuk simak.

Menggunakan Array

```
$pelajaran = ['Matematika','Produktif','English'];
return view('sekolah',['pelajaran'=>$pelajaran]);
```

atau

```
$data['pelajaran'] = ['Matematika','Produktif','English'];
return view('sekolah',$data);
```

Menggunakan Fungsi with

```
// Using conventional approach
$view = view('greeting')->with('name', 'Victoria');
// Using Magic Methods
$view = view('greeting')->withName('Victoria');
```

Menggunakan fungsi compact

```
$pelajaran = ['Matematika','Produktif','English'];
return view('sekolah',compact('pelajaran'));
```

Migration dan Operasi Database

Buat Aplikasi Laravel baru

Pada kasus kali ini kita akan membuat aplikasi CRUD data siswa sederhana. Pertama, kita masuk ke direktori `/var/www/html`

```
cd /var/www/html
```

Kemudian kita buat project laravel baru

```
composer create-project laravel/laravel laravel 5.0 --prefer-dist
```

Setelah proses instalasi project baru laravel selesai, kita masuk ke folder laravel

```
cd laravel
```

Pasang dependency html

Karena di Laravel 5 ini package untuk html sudah di remove, maka kita akan tambahkan secara manual melalui composer dengan perintah berikut

```
composer require illuminate/html
```

Kemudian kita update melalui composer

```
composer update
```

Sekarang buka file app.php kemudian tambahkan sc[ript berikut di bagian providers

```
'Illuminate\Html\HtmlServiceProvider',
```

Dan script berikut pada bagian aliases

```
'HTML'  => 'Illuminate\Html\HtmlFacade',
'Form'  => 'Illuminate\Html\FormFacade',
```

Setelah itu kita dump autoload dengan composer

```
composer dump-autoload
```

Buat database baru

Kemudian kita buat databasenya melalui mysql

```
create database datasiswa;
```

Konfigurasi .env

Buka file .env kemudian ubah bagian dibawah ini sesuai konfigurasi database kita masing-masing

```
DB_HOST=localhost
DB_DATABASE=datasiswa
DB_USERNAME=root
DB_PASSWORD=password
```

Migration

Selanjutnya buat file migration dengan perintah berikut di terminal

```
php artisan make:migration create_table_siswa --create="siswas"
```

Kemudian kita buka file migration create_table_siswa yang sudah kita buat tadi di `/database/migrations` kemudian kita buat seperti dibawah ini

```
<?php

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

```
class CreateTableSiswa extends Migration {

 /**
  * Run the migrations.
  *
  * @return void
  */
 public function up()
 {
        Schema::create('siswas', function(Blueprint $table)
        {
                $table->increments('id');
                $table->string('nama');
                $table->string('jenis_kelamin');
                $table->date('tanggal_lahir');
                $table->timestamps();
        });
 }
 /**
  * Reverse the migrations.
  *
  * @return void
  */
 public function down()
 {
        Schema::drop('siswas');
 }
}
```

Kemudian kita jalankan perintah migrate di terminal

```
php artisan migrate
```

Buat model

Selanjutnya kita buat model melalui artisan dengan script berikut

```
php artisan make:model Siswa
```

Konfigurasi routes.php

Tambahkan script berikut pada file routes.php

```
Route::bind('siswa',function ($siswa)
{
 return App\Siswa::find($siswa);
});
Route::resource('siswa','SiswaController');
```

Buat controller

```
php artisan make:controller SiswaController
```

Kita buka file SiswaController.php kemudian isi dengan script berikut

```
<?php namespace App\Http\Controllers;

use App\Http\Requests;
use App\Http\Controllers\Controller;
use App\Siswa as Siswa;

use Illuminate\Http\Request;

class SiswaController extends Controller {


 /**
  * Display a listing of the resource.
  *
  * @return Response
  */
 public function index(Siswa $siswa)
 {

        $siswas = $siswa->get();

        return view('siswa.list',compact('siswas'));
 }

 /**
  * Show the form for creating a new resource.
  *
  * @return Response
  */
 public function create(Siswa $siswa)
 {
        return view('siswa.add',compact('siswa'));
 }

 /**
  * Store a newly created resource in storage.
  *
  * @return Response
  */
 public function store(Request $req, Siswa $siswa)
 {
        $siswas = new $siswa;
        $siswas->nama = $req->nama;
        $siswas->jenis_kelamin = $req->jenis_kelamin;
        $siswas->tanggal_lahir =
date_format(date_create($req->tanggal_lahir),"Y-m-d");
```

```
        $siswas->save();

        return redirect('siswa');
 }
 /**
  * Display the specified resource.
  *
  * @param  int  $id
  * @return Response
  */
 public function show(Siswa $siswa)
 {
        return view('siswa.detail',compact('siswa'));
 }

 /**
  * Show the form for editing the specified resource.
  *
  * @param  int  $id
  * @return Response
  */
 public function edit(Siswa $siswa)
 {
        return view('siswa.edit',compact('siswa'));
 }
 /**
  * Update the specified resource in storage.
  *
  * @param  int  $id
  * @return Response
  */
 public function update(Request $req, Siswa $siswa)
 {
        $siswa->nama = $req->nama;
        $siswa->jenis_kelamin = $req->jenis_kelamin;
        $siswa->tanggal_lahir =
date_format(date_create($req->tanggal_lahir),"Y-m-d");
        $siswa->save();

        return redirect('siswa');
 }
 /**
  * Remove the specified resource from storage.
  *
  * @param  int  $id
  * @return Response
  */
 public function destroy(Siswa $siswa)
 {
        $siswa->delete();

        return redirect('siswa');
 }

}
```

Buat views

Buat folder baru di /resources/views bernama siswa kemudian buat file form.blade.php didalamnya, kemudian isikan dengan script berikut

```
<div class="form-group">
 {!! Form::label('nama') !!}
 {!! Form::text('nama',null,['class'=>'form-control']) !!}
</div>

<div class="form-group">
 {!! Form::label('jenis_kelamin') !!}
 {!!
Form::text('jenis_kelamin',null,['class'=>'form-control']) !!}
</div>

<div class="form-group">
 {!! Form::label('tanggal_lahir') !!}
 {!!
Form::text('tanggal_lahir',null,['class'=>'form-control']) !!}
</div>

<div class="form-group">
 {!! Form::submit('Send',null,['class'=>'form-control']) !!}
</div>
```

Kemudian buat file add.blade.php dan isikan dengan script berikut

```
@extends('app')

@section('content')

<div class="container">

{!! link_to_route("siswa.index","Siswa") !!}

<hr>
 {!! Form::model($siswa,
['route'=>['siswa.store',$siswa->id],'method'=>'POST']) !!}

        @include ('siswa.form')

 {!! Form::close() !!}

</div>

@endsection
```

Selanjutnya buat file detail.blade.php kemudian isikan script berikut

```
@extends('app')

@section('content')

<div class="container">
{!! link_to_route("siswa.index","Siswa") !!}

<hr>

{{ $siswa->nama }}<br>
{{ $siswa->jenis_kelamin }}<br>
{{ $siswa->tanggal_lahir }}

<hr>

{!! link_to_route("siswa.edit","Edit",$siswa->id) !!}

{!!
Form::open(['method'=>'DELETE','route'=>['siswa.destroy',$siswa-
>id]]) !!}

 {!! Form::submit('Delete',['style'=>'background-color:
transparent;border:0px;padding:0px;color:
#337ab7;text-decoration: none;','onclick'=>'return confirm("Are
you sure?")']) !!}

{!! Form::close() !!}

</div>

@endsection
```

Selanjutnya buat file edit.blade.php dan isikan script berikut

```
@extends('app')
@section('content')
<div class="container">

{!! link_to_route("siswa.index","Siswa") !!}
<hr>

 {!! Form::model($siswa,
['route'=>['siswa.update',$siswa->id],'method'=>'PATCH']) !!}

        @include ('siswa.form')

 {!! Form::close() !!}

</div>
@endsection
```

Kemudian kita buat file list.blade.php dan isikan script berikut

```
@extends('app')

@section('content')

<div class="container">

{!! link_to_route("siswa.index","Siswa") !!}

<hr>

{!! link_to_route("siswa.create","New") !!}
<ul>

@foreach ($siswas as $data)
       <li>
       {!! link_to_route("siswa.show",$data->nama,$data->id) !!}
       </li>
@endforeach

</ul>

</div>

@endsection
```