

## BAB III

### Model View Controller (MVC)

#### A. Pengertian

Selama ini anda membangun aplikasi menggunakan PHP menyatukan seluruh fungsinya kedalam satu file meliputi tampilan, proses database dan pengaturan perpindahan page. Dengan semakin berkembangnya project aplikasi maka model pengembangan dengan menyatukan seluruh fungsi tersebut akan semakin merepotkan baik dalam maintenance maupun dalam pekerjaan team work.

Untuk mengantisipasi keterbatasan tersebut maka sebaiknya arsitektur pengembangan aplikasi yang memisahkan dan mengelompokkan beberapa kode sesuai dengan fungsinya dimana seluruh fungsi yang mengatur ke dalam tampilan, proses database dan pengaturan perpindahan page.

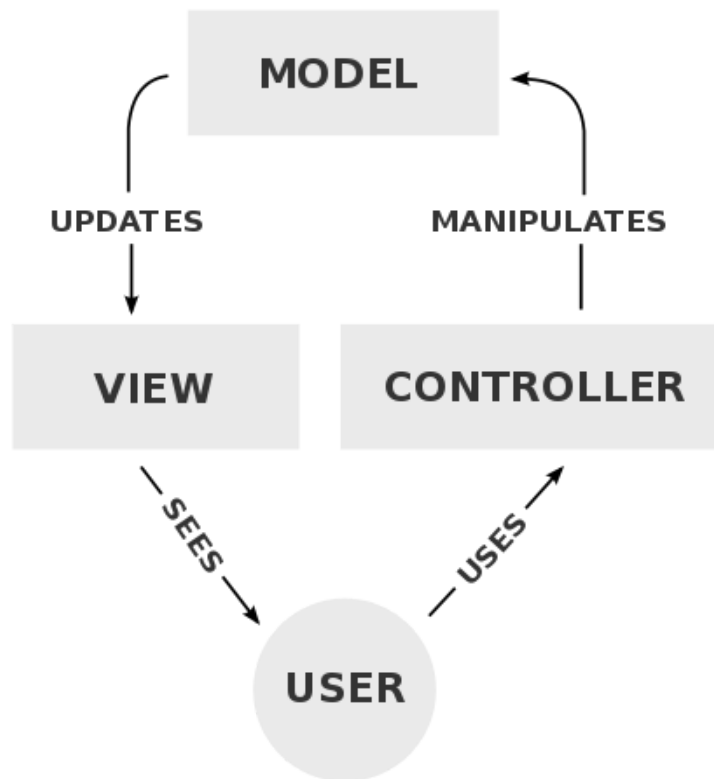
Arsitektur yang diharapkan adalah arsitektur pengembangan yang bisa membagi aplikasi ke dalam bagian fungsional yaitu model, view, dan controller. Arsitektur seperti ini sering disebut arsitektur pengembangan MVC (Model, View, and Controller). Adapun maksud dari ketiga fungsi tersebut adalah :

- a. **Model** adalah kode-kode untuk model bisnis dan data. biasanya berhubungan langsung dengan database untuk memanipulasi data (insert, update, delete, search), menangani validasi dari bagian controller, namun tidak dapat berhubungan langsung dengan bagian view.
- b. **View** merupakan bagian yang menangani presentation logic. berisi kode-kode untuk tampilan.
- c. **Controller** merupakan bagian yang mengatur hubungan antara bagian model dan bagian view, controller berfungsi untuk menerima request dan data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.

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).

Pengembangan model MVC pun bisa dilakukan tanpa framework atau bisa dengan framework, dan ini adalah hal yang umum pada saat ini yaitu menggunakan framework yang mendukung MVC.

Berikut adalah skema kerja aplikasi yang berbasis Model View dan Controller baik dengan framework maupun tanpa framework.



## B. Jenis-jenis MVC

Seiring dengan perkembangan teknologi web jenis-jenis MVC pun berkembang pesat apalagi semenjak lahirnya javascript untuk server side scripting, secara garis besar jenis-jenis MVC adalah sebagai berikut :

1. **Server Side MVC**, Server Side MVC biasa terjadi pada aplikasi web tradisional, yang tidak melibatkan client side seperti *Javascript*, *Java applet*, *Flash*, dan lain-lain. Server Side MVC menyerahkan keseluruhan proses bisnis pada server, aplikasi pada sisi pengguna hanya dapat menerima. MVC jenis ini kadang-kadang disebut juga dengan nama *Thin Client*.
2. **Mixed Client Side and Server Side MVC**, Pada *Mixed Client Side and Server Side MVC* 1 client tidak menggunakan model sebagai jembatan untuk melakukan komunikasi

pada server, dibandingkan dengan Server Side MVC, arsitektur ini memiliki tingkat kompleksitas yang lebih tinggi karena lebih banyak komponen yang terlibat. Untuk selanjutnya arsitektur ini disebut, dengan Mixed MVC 1. Pada *Mixed Client Side and Server Side MVC* 2, client menggunakan model sebagai jembatan untuk melakukan komunikasi pada server, dibandingkan dengan arsitektur MVC yang lain, arsitektur ini memiliki tingkat kompleksitas yang paling tinggi karena lebih banyak komponen yang terlibat, sehingga membutuhkan sumber daya yang lebih besar pula. Untuk selanjutnya arsitektur ini disebut dengan *Mixed MVC* 2.

3. **Rich Internet Application MVC**, *Application MVC Rich Internet Application* (RIA) disebut juga dengan nama *Fat Client*, merupakan aplikasi web yang memiliki kemampuan dan fungsi hampir seperti aplikasi desktop. RIA pada sisi client, memiliki mesin untuk mengambil data yang berada pada server, sehingga pada client terdapat bagian MVC sendiri dan hanya membutuhkan bagian model pada sisi server.

### C. Jenis-jenis Framework PHP Berbasis MVC

Para web developer modern, khususnya para web developer yang mengembangkan situs ataupun aplikasi menggunakan PHP akan menggunakan framework tertentu untuk membantu pekerjaan mereka menjadi lebih mudah dan efisien. Berikut adalah framework php modern yang menggunakan Model Arsitektur MVC :

#### 1. Laravel

Laravel merupakan framework php yang menggunakan konsep MVC (Model-View-Controller) yang dikembangkan oleh Taylor Otwell. PHP framework ini tergolong masih baru. PHP dirilis tahun 2011, dimana framework ini baru berumur sembilan tahun, terhitung dari tahun Laravel dirilis. Akan tetapi, PHP framework ini diklaim menjadi framework yang cukup populer di kalangan web developer. Laravel juga memiliki komunitas yang cukup banyak dan besar, termasuk di Indonesia. Selain itu, terdapat berbagai tutorial pada situs resminya apabila Anda ingin mempelajari laravel.

#### 2. CodeIgniter

Salah satu PHP framework yang menggunakan juga mengusung konsep MVC adalah CodeIgniter. Framework ini dirilis tahun 2006, dimana framework php ini tergolong memiliki umur yang cukup lama. Apabila dihitung dari tahun rilis Codeigniter, framework PHP ini telah berumur 12 tahun. Maka dari itu, wajar apabila php framework ini cukup

banyak digunakan dan dikenal oleh para web developer. CodeIgniter memiliki proses instalasi yang cukup mudah. Selain itu, CodeIgniter dapat berjalan pada sebagian besar platform hosting (shared & dedicated hosting). CodeIgniter versi 3 tidak termasuk kedalam kelompok php modern, namun sampai saat ini codeigniter versi 4 sedang dikembangkan mengukung php modern.

### **3. Yii 2**

Hampir sama seperti framework yang telah disebutkan sebelumnya, Yii merupakan framework open source yang mengukung konsep MVC. Yii 2 juga mengukung konsep DRY (Don't Repeat Yourself), yang menyediakan kode-kode dasar bagi para web developer untuk membantu mereka dalam mengembangkan situs. Lalu, Yii 2 juga dapat terintegrasi dengan baik pada jQuery dan dilengkapi dengan fitur AJAX enabled.

### **4. Symfony**

Symfony merupakan framework yang diciptakan untuk pemeliharaan dan pembaharuan suatu situs. Sebab, framework PHP ini dapat menggantikan kode-kode yang digunakan secara berulang kali pada suatu situs. Selain itu, komponen yang dimiliki Symfoni membuat Anda dapat menggunakan library PHP kembali. Apabila Anda ingin melihat hasil akhir dari suatu proyek yang dikembangkan menggunakan Symfoni, Anda dapat berkunjung ke situs resminya. Disana, tertera proyek-proyek besar yang dikembangkan menggunakan Symfony dengan hasil akhir yang memuaskan.

## BAB IV

### PHP Framework

#### A. Pengertian

Framework PHP adalah platform dasar yang memungkinkan kita untuk mengembangkan aplikasi web. Dengan kata lain, framework menyediakan struktur standar. Dengan menggunakan PHP Framework, Anda akan menghemat banyak waktu, menghentikan kebutuhan untuk menghasilkan kode berulang, dan Anda akan dapat membangun aplikasi dengan cepat (*Rapid Application Development*). Tanpa Framework PHP akan jauh lebih sulit untuk menghasilkan aplikasi karena Anda harus berulang kali mengetik kode yang sama untuk fungsi yang sama. Anda juga harus menjalankan koneksi antara database Anda dan aplikasi apa pun yang Anda kembangkan dari nol. Sementara jika menggunakan PHP Framework hal itu tidak perlu anda lakukan dari nol semua sudah dipersiapkan anda tinggal melakukan penyesuaian terhadap framework yang anda gunakan.

Framework PHP beroperasi pada arsitektur Model View Controller (MVC). MVC adalah pola arsitektur yang ditampilkan dalam berbagai bahasa pemrograman populer yang memecah logika domain Anda dari antarmuka pengguna Anda. Logika domain adalah fungsi yang menangani pertukaran informasi antara database Anda dan antarmuka pengguna Anda. Karenanya, Anda dapat memodifikasi logika domain dan yang paling penting bagi desainer, antarmuka pengguna secara terpisah. Ini menghilangkan banyak kebingungan dan menyederhanakan seluruh proses perkembangan. Ketika kita merujuk ke MVC kita biasanya melihatnya sebagai ini: M adalah singkatan dari data mentah, V (tampilan / antarmuka pengguna) mewakili apa yang sebenarnya dilihat, dan C (pengontrol) sebenarnya adalah logika domain seperti yang terlihat di atas. Setelah Anda dapat memahami bagaimana MVC bekerja, maka Kerangka PHP menjadi jauh lebih jelas dan lebih mudah digunakan. Untuk jelasnya tentang MVC silahkan baca kembali bab sebelumnya tentang MVC.

#### B. Hal-hal Yang Harus Diperhatikan Dalam Memilih Framework

Dari sekian banyak php framework manakah yang seharusnya kita gunakan ? untuk menjawab pertanyaan seperti ini tergantung dari sudut individu programmer, namun secara umum tidak setiap framework PHP menawarkan dukungan yang sama untuk basis data, komunitas, dan panduan pengguna yang mudah diikuti. Hal Itu mungkin baik-baik saja jika

Anda mencari sesuatu yang sangat sederhana. Namun, jika Anda menemukan framework PHP yang Anda sukai, harus ada berbagai opsi dan keuntungan yang menyertainya. Sebaiknya perhatikan hal-hal dibawah ini :

1. Dukungan Basis Data

Dukungan basis data sangat penting. Misalnya, CodeIgniter mendukung MySQL, Oracle, dan SQLite, sedangkan Framework Kohana tidak mendukung Oracle atau SQLite. Bergantung pada basis data mana yang Anda pilih atau pilih untuk proyek Anda. Saat Laravel mendukung empat database yaitu MySQL, Postgres, SQLite, and SQL Server. Anda juga perlu mempertimbangkan apakah server basis data Anda mendukung jenis basis data ini.

2. Dukungan Komunitas

Framework Anda harus memiliki komunitas yang kuat, tidak hanya dalam hal banyaknya anggota tetapi juga dalam aktivitas dan bantuan. Bahkan jika itu adalah komunitas kecil, selama Anda dapat menemukan dukungan, maka itu adalah poin plus bagi framework tersebut.

3. Dukungan Dokumentasi

Pastikan bahwa Framework PHP Anda memiliki dokumentasi yang baik yang selalu diperbarui, dan panduan pengguna ini relatif mudah diikuti.

4. Dukungan Terhadap Arsitektur MVC

Framework harus mendukung arsitektur Model View Controller. Sebagian besar kerangka framework yang akan Anda temukan juga menawarkan library, plugin, help, dan extension. Ada baiknya untuk menemukan kerangka kerja yang memiliki setidaknya dua opsi ini.

## **BAB V**

### **Laravel PHP Framework**

#### **A. Sejarah**

Laravel adalah sebuah framework PHP yang dirilis dibawah lisensi MIT, dibangun dengan konsep MVC (model view controller). Laravel adalah pengembangan website berbasis MVP yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan, dan untuk meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu.

Sejarah framework Laravel dibuat oleh Taylor Otwell, proyek Laravel dimulai pada April 2011. Awal mula proyek ini dibuat, karena Otwell sendiri tidak menemukan framework yang up-to-date dengan versi PHP. mengembangkan framework yang sudah ada juga bukan merupakan ide yang bagus, karena keterbatasan sumber daya. Dikarenakan beberapa keterbatasan tersebut, Otwell membuat sendiri framework dengan nama Laravel. Oleh karena itu, Laravel mensyaratkan PHP versi 5.3 ke atas.

Pada Agustus 2009, PHP 5.3 resmi dirilis. Dalam rilis tersebut, PHP 5.3 sudah support dengan object oriented yang lebih baik. Framework yang support dengan PHP versi 5.3 adalah Symfony, Zend, Kohana, Lithium dan CodeIgniter.

CodeIgniter mungkin framework PHP yang paling terkenal pada saat itu. Developer framework PHP menyukainya karena dokumentasi dari berbagai forum yang banyak dan source code yang sederhana. Setiap programmer PHP dengan cepat bisa mulai membuat aplikasi dengan framework tersebut karena komunitasnya besar dan dukungan besar dari penciptanya.

Namun pada tahun 2011, CodeIgniter memiliki kekurangan seperti yang diungkapkan oleh Taylor Otwell, Creator Laravel menyebutkan ada beberapa fitur fungsional yang penting yang tidak support, seperti kotak autentikasi dan routing. Oleh karena itu, Laravel versi beta 1 dirilis pada tanggal 9 Juni 2011 untuk mengisi fungsi yang hilang. Menurut pencipta Laravel itu (Taylor Otwell), Laravel versi 1 dirilis pada Juni 2011 hanya untuk menambah kekurangan yang ada didalam framework CodeIgniter PHP.

##### **a. Release Laravel 1**

Dimulai dengan rilis pertama, fitur Laravel dibangun dengan Autentikasi, Eloquent ORM (Object Relational Mapping) untuk operasi database, localization, model dan relationship, mekanisme routing yang sederhana, caching, session, views, module dan library, HTML, dsb. Bahkan pada rilis pertama, Laravel sudah memiliki beberapa fungsi mengesankan.

pada saat itu Laravel versi ini belum berbasis MVC (Model View Controller), tetapi developer menyukai karena sintaks yang friendly dan potensi framework baru ini yang begitu menjanjikan. Dalam bulan-bulan berikutnya, Taylor menambahkan method validasi, pagination, paket command line installer, ekspansi Eloquent ORM (Object Relational Mapping), dan termasuk beberapa ratus unit testing untuk komponen framework. Laravel versi 1 ke versi berikutnya dalam kurun waktu kurang dari enam bulan.

b. Release Laravel 2

Laravel versi 2 dirilis ke developer pada 24 November 2011, upgrade beberapa fitur diantaranya dukungan controller, engine template dan penggunaan invers. Dengan penambahan fitur controller ini, maka Laravel versi 2 ini sudah resmi menjadi framework yang berbasis MVC. Kurang dari dua bulan kemudian resmi Laravel 3 dirilis.

c. Release Laravel 3

Pada 22 Februari 2012, Laravel 3 dirilis, memfokuskan pada unit test integration, artisan command line interface, database migration, session driver dan database driver.

Forum Laravel terus menerus menerangi pengguna framework ini. Laravel 3 dirilis secara stabil untuk beberapa waktu. Sekitar 5 bulan setelah dirilis, creator Laravel memutuskan untuk menulis ulang framework dari awal sebagai satu set paket yang didistribusikan melalui composer. Kemudian barulah dirilis Laravel 4, upgrade signifikan yang menampilkan arsitektur yang berbeda dari inti framework.

d. Release Laravel 4

Tampaknya ada versi terbaru dari Laravel setiap beberapa bulan. Laravel 4 secara resmi dirilis satu tahun dan 3 bulan setelah rilis versi 3 tepat pada tanggal 28 Mei 2013. Beberapa developer menyebutnya “terlalu cepat” update dari versi satu ke versi yang lain, karena mereka harus bermigrasi ke versi baru dan kadang-kadang itu hanya tidak mungkin dengan aplikasi besar yang sudah dibangun pada arsitektur



sebelumnya. Masyarakat meminta untuk lebih stabil, beberapa fitur baru dan unit testing yang lebih baik dari komponen Laravel itu.

Laravel 4 ditulis ulang dari bawah ke atas sebagai kumpulan komponen (atau paket) yang terintegrasi dengan satu sama lain untuk membuat framework yang stabil. Pengelolaan komponen ini dilakukan melalui "Composer" yang disebut sebagai PHP dependency manager. Laravel 4 memiliki fitur yang ada di versi lain atau bahkan framework yang telah ada sebelumnya, seperti database seeding, message queues, built-in mailer, fitur Eloquent ORM, soft delete, dan bahkan lebih dari itu. Berbeda dengan versi sebelumnya, Laravel 4 ini akan ada jadwal rilis secara teratur setiap 6 bulan untuk update (patch dan perbaikan bug). Dengan unit test yang meliputi 100% dari fungsi framework tersebut, Laravel 4 ini akan menjanjikan untuk menjadi stabil dan mudah di update secara online melalui composer.

## **B. Instalasi Software Pendukung**

### **a. PHP Dependency Manager**

*Composer* adalah *dependency manager* untuk bahasa pemrograman PHP, bagaimana maksudnya? *dependency* adalah kebutuhan atau ketergantungan, maksudnya setiap kita membuat proyek pasti kita akan menemui saat dimana kita membutuhkan library atau package untuk fungsi fungsi yang akan kita buat dalam proyek kita.

Jika sebelum ada composer ini saya masih ingat saat membutuhkan library atau package maka kita harus download library-nya kemudian kita ekstrak filenya dan kemudian kita masukkan folder hasil ekstrak tadi ke dalam proyek yang sedang kita buat. Nah proses tersebut tentunya memakan banyak waktu kalau kita butuh banyak library untuk proyek kita.

Dengan composer proses tersebut bisa menjadi lebih mudah & simpel, jadi composer menggantikan / membuat proses tersebut menjadi 'otomatis' sehingga kita tidak perlu download - ekstrak - salin ke proyek, tapi kita cukup jalankan composer pada proyek yang sedang kita buat.

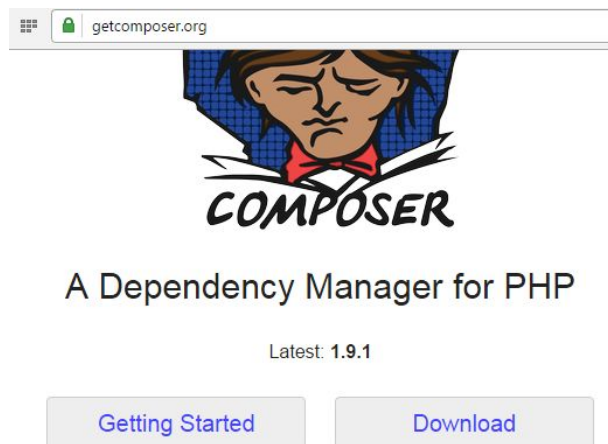
Sebelum menginstal composer di windows sebaiknya diperhatikan hal-hal seperti berikut :

1. Arsitektur prosesor, sebaiknya menggunakan prosesor dengan arsitektur 64 bit baik.
2. PHP harus menggunakan minimal versi 5.5 atau versi 7 / terbaru lebih baik
3. Sistem sebaiknya menggunakan OS 64 bit, karena OS dengan 32 bit sudah sangat berkurang untuk dukungan aplikasi yang akan digunakan
4. Jika terpaksa menggunakan arsitektur OS 32 bit maka setidaknya sistem operasi untuk windows minimal windows 7 SP 1

- Instalasi Composer

Jika persyaratan diatas sudah terpenuhi maka langkah-selanjutnya mari kita instal composer.

1. Download composer di url <http://getcomposer.org>, kemudian klik tombol download



*Gambar 5.1*  
*Web Site Composer.org*

2. Untuk pengguna OS Windows silahkan click link **Composer-Setup.exe**

# Download Composer Latest: v1.9.1

## Windows Installer

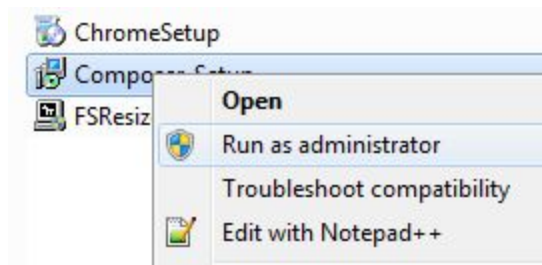
The installer will download composer for you and set up. You can simply call `composer` from any directory.

Download and run [Composer-Setup.exe](#) - it will install it

*Gambar 5.2*

*Link download composer-setup.exe*

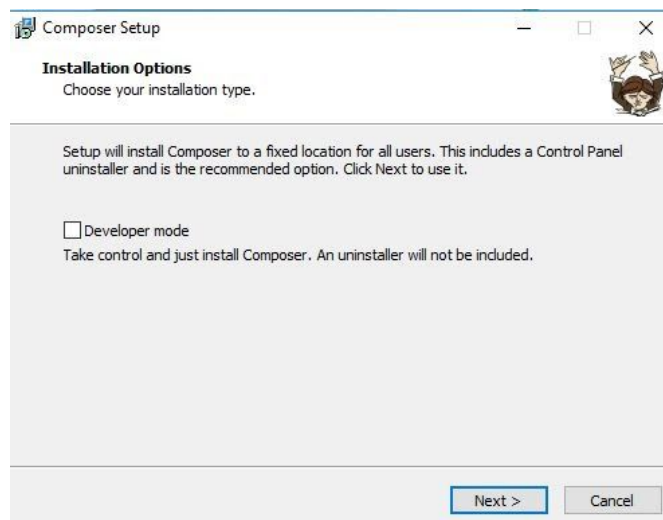
3. Setelah selesai di-download, silahkan instal dengan cara klik-kanan file `Composer-Setup.exe` kemudian Klik Run Administrator



*Gambar 5.3*

*Install Composer dengan role Administrator*

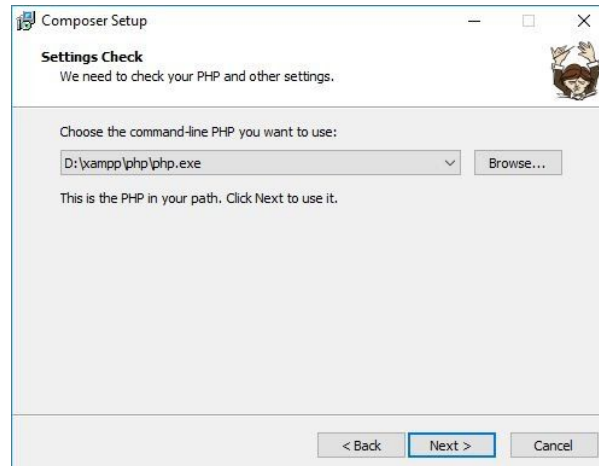
4. Pada langkah berikutnya jangan men-ceklist Developer mode, lanjutkan dengan menekan tombol NEXT



*Gambar 5.4*

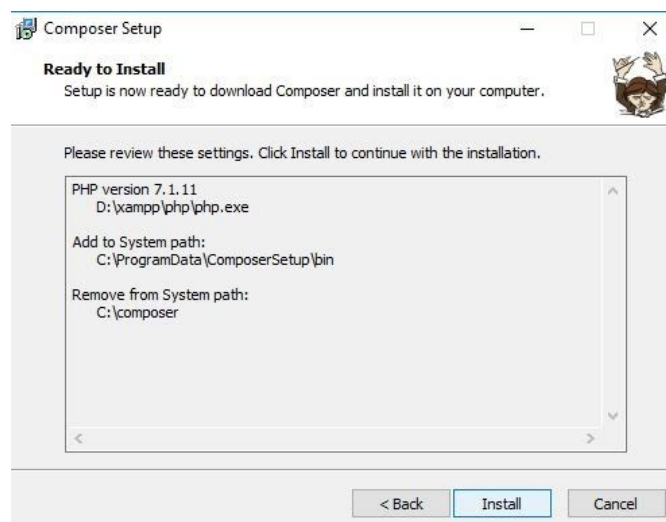
*Pemilihan mode instalasi*

5. Pada tahap ini akan mengecek apakah di computer kita terinstal php ? jika ya maka path url ke file php.exe maka text box command line php otomatis terisi, jika memiliki dua versi php yang berbeda silahkan isi manual dengan menekan tombol browse dan cari file php.exe yang sesuai, jika telah yakin klik tombol NEXT



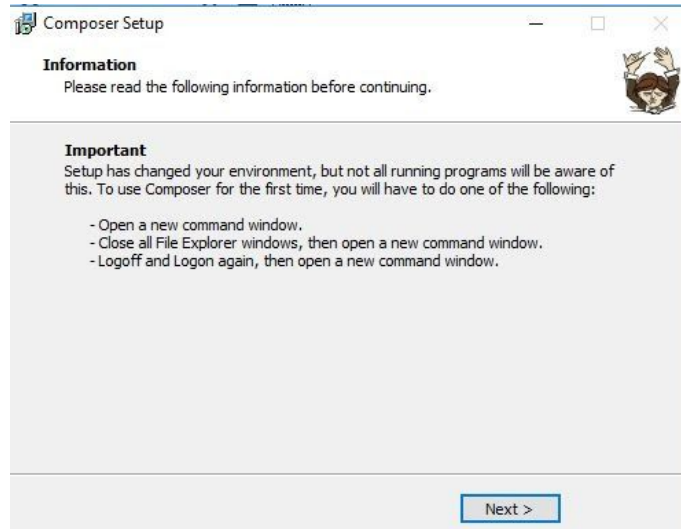
*Gambar 5.5  
System checking instalasi*

6. Pada tahap berikutnya mengecek kesiapan instalasi, jika sudah benar maka klik tombol install dan tunggu sampai proses selesai. **Harap Diingat !** proses instalasi composer membutuhkan akses internet jika tidak maka proses instalasi akan digagalkan !



*Gambar 5.6  
Review persiapan instalasi*

7. Pada saat selesai tahap instalasi, akan diberikan informasi sederhana tata cara penggunaan dan menjalankan composer pertama kali seperti tampak pada gambar dibawah ini !



*Gambar 5.7*  
*Web Site Composer.org*

8. Pada tahap terakhir klik tombol Finish untuk menyelesaikan

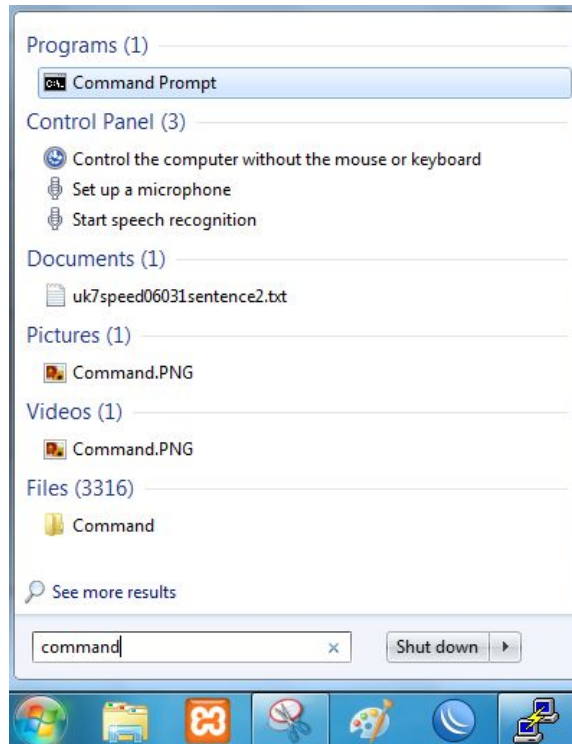


*Gambar 5.8*  
*Instalasi Composer Selesai*

- MenjalankanComposer

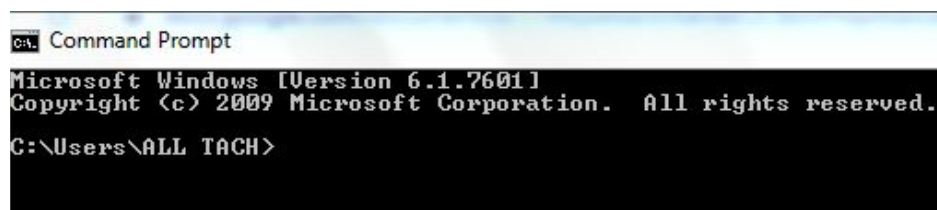
Setelah selesai instalasi composer tahap berikutnya kita coba jalankan composer, berikut adalah langkah-langkah menjalankan composer :

1. Klik **start**
2. Ketik **command**, kemudian klik **Command Prompt**



*Gambar 5.9  
Menjalankan Command Prompt*

3. Maka dilayar akan tampil command prompt seperti tampak pada gambar dibawah ini



*Gambar 5.10  
Menjalankan Command Prompt Windows*

4. Untuk menguji apakah composer bisa digunakan ketik **composer** pada **command prompt** tersebut kemudian **tekan enter**, jika tampil serangkaian informasi seperti tampak pada gambar di halaman berikutnya berarti proses instalasi composer telah berhasil.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ALL TACH>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 questions
      --profile            Display timing and memory usage
      --no-plugins         Whether to disable plugins.
```

Gambar 5.11  
Menjalankan Composer di Command Prompt

b. Code Editor

Code editor yang digunakan selama ini adalah notepad++ untuk pengembangan aplikasi menggunakan Framework PHP modern seperti laravel notepad++ memiliki beberapa kekurangan diantaranya :

1. Kesulitan dalam membuka file antara folder
2. Tidak terintegrasi nya terminal/command prompt didalam code editor
3. Tidak memiliki fitur auto complete
4. Tidak memiliki fitur untuk merapikan baris code

Banyak sekali code editor yang mampu menanggapi kelemahan-kelemahan diatas diantaranya :

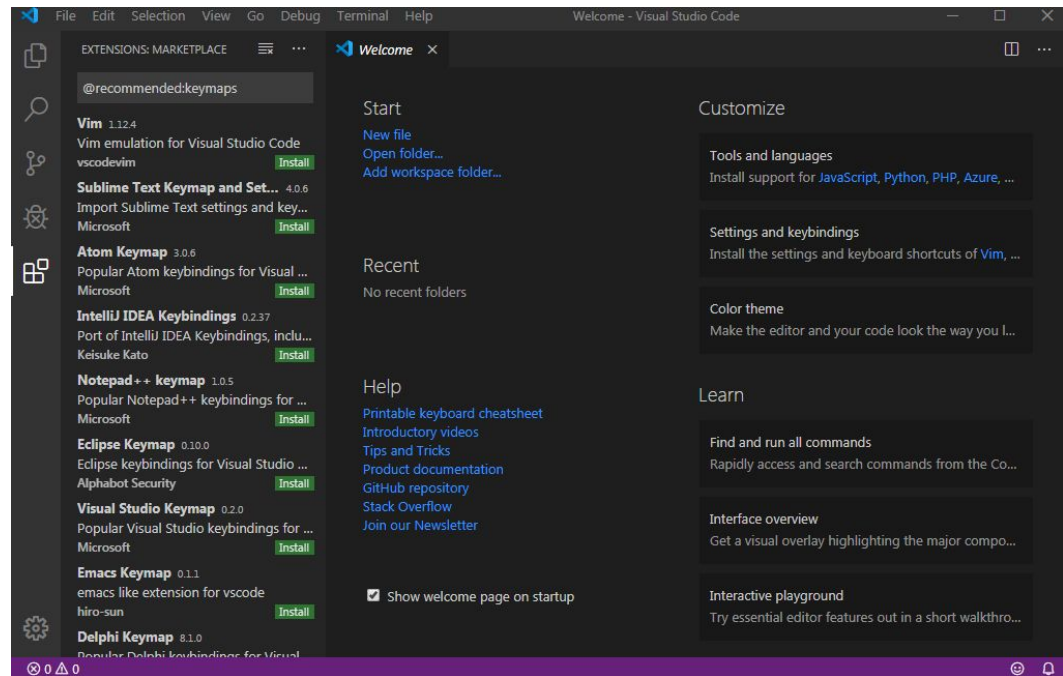
1. Visual studio code
2. Atom.io
3. Sublime Text
4. Bracket, dll

Untuk pelajaran kali ini editor manakah yang akan digunakan ? tidak ada keharusan untuk memilih salah satu silahkan saja dicoba yang mana yang anda suka, namun untuk keseragaman pada materi ini akan digunakan code editor Visual Studio Code, berikut langkah-langkah instalasi :

1. Download code editor visual studio code di alamat berikut :  
**<https://code.visualstudio.com/download>**, Sebagai catatan untuk

mendownload di halaman tersebut sistem operasi anda harus menggunakan arsitektur 64 bit, sedangkan untuk yang 32 bit bisa di searching di google dengan kata kunci “**Download Visual Studio Code 32 bit For Windows**”

2. Lakukan instalasi dengan cara double click file hasil download
3. Ikuti setiap langkah yang ditunjukkan dalam layar monitor
4. Apabila telah selesai silahkan jalankan aplikasi nya dengan meng-klik **Start -> All Programs -> Visual Studio Code -> Visual Studio Code**



Gambar 5.12  
Tampilan Visual Studio Code

### C. Instalasi Laravel Framework

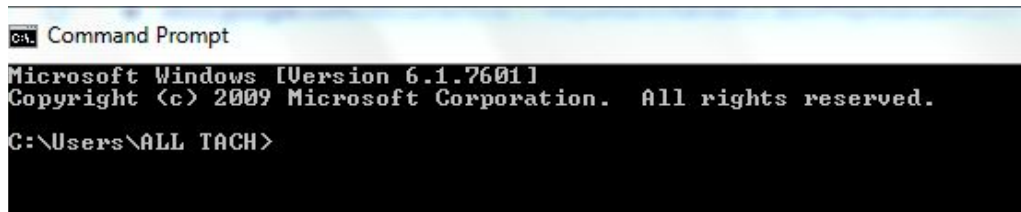
Untuk instalasi framework laravel berbeda php prosedural dimana laravel tidak membutuhkan web server apache karena di dalam framework laravel sudah tersedia *built in web server* (Web server bawaan).

Untuk instalasi framework PHP laravel, mutlak diperlukan akses internet karena proses instalasi pada dasarnya adalah men-download seluruh file-file yang diperlukan oleh laravel kedalam folder project yang didefinisikan saat membuat project baru, maka jangan heran apabila file framework laravel akan memiliki ukuran yang cukup besar sebelum kita benar-benar membuat project berbasis laravel.

Berikut adalah langkah-langkah membuat project laravel :



1. Jalankan command prompt di windows atau terminal di linux



*Gambar 5.13  
Command Prompt Windows*

2. Bila perlu pindah drive untuk menyimpan file-file project laravel misal di drive D atau E dengan mengetik huruf drive diikuti tanda titik dua kemudian tekan enter

Misal :

```
C:\user\rpl>E: (tekan enter)
```

3. Maka command prompt akan berubah menjadi seperti berikut :

```
E:\>
```

4. Jika akan pindah ke drive D cukup ketik D diikuti tanda titik dua kemudian tekan enter

```
E:\>D: (tekan enter)
```

5. Maka command prompt akan berubah menjadi seperti berikut :

```
E:\>_
```

6. Untuk menyamakan persepsi dalam latihan ini mari kita gunakan Drive D untuk menyimpan project laravel

7. Buat folder baru di drive D dengan nama folder **project\_laravel**,

```
D:\>md project_laravel (tekan enter)
```

8. Masuk ke folder project\_laravel

```
D:\>cd project_laravel (tekan enter)
```

9. Jalankan perintah composer untuk membuat project baru dengan laravel

```
D:\project_laravel>composer create-project --prefer-dist  
laravel/laravel latihan_laravel
```

### **Keterangan**

Composer	: Menjalankan composer
Create-project	: Membuat project baru
--prefer-dist	: Mendownload versi stabil / terbaru

laravel/laravel : Mendownload laravel  
Latihan\_laravel : Nama project (nama project tidak boleh menggunakan spasi atau tanda minus sebagai penghubung)

10. Tunggu proses instalasi sampai selesai (rata-rata membutuhkan waktu lima menit, tergantung koneksi internet yang digunakan)

#### D. Menjalankan Framework Laravel

Setelah selesai membuat project dengan perintah baris diatas, selanjutnya bagaimana cara menjalankannya ? berikut adalah langkah-langkah menjalankan framework laravel

1. Masuk ke command prompt atau bua tab terminal di Visual Code Editor
2. Masuk ke project yang telah dibuat

```
CD D:\project_laravel\latihan_laravel (tekan enter)
```

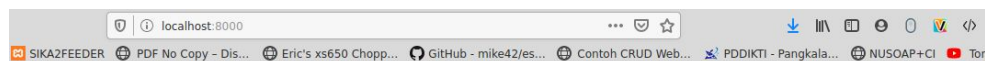
3. Ketik perintah berikut untuk menjalankan laravel

```
D:\project_laravel\latihan_laravel>php artisan serve (tekan enter)
```

```
laravel development server started: http://127.0.0.1:8000
```

```
[Tue Jan 7 10:16:02 2020] 127.0.0.1:54286 [200]: /favicon.ico
```

4. Jalankan browser ketik alamat 127.0.0.1:8000 pada address bar, apabila sukses maka di browser akan tampil layar perdana tampilan laravel yang menandakan anda telah berhasil menginstal laravel



Laravel

DOCS   LARACASTS   NEWS   BLOG   NOVA   FORGE   VAPOR   GITHUB

**Gambar 5.14**  
*Tampilan Awal Laravel*

#### E. Mengecek Versi Laravel Yang Digunakan

Pada saat kita memberikan perintah create-project untuk membuat project laravel, kita tidak tahu laravel versi berapa yang terinstall, untuk keperluan tertentu mengetahui versi laravel adalah suatu hal yang harus agar aplikasi kita bisa menggunakan library-library lain yang sesuai dengan laravel yang kita install. Berikut adalah langkah-langkah untuk mengetahui versi laravel yang digunakan :

- a. Jalankan command, klik start
- b. Ketik command, klik **Command Prompt**
- c. Masuk ke drive dimana project laravel anda terinstall, misal di drive d

```
C:\users\nama_user>D: (tekan enter)
```

- d. Masuk ke folder dimana project laravel disimpan misal di folder **project\_laravel**

```
D:\>cd project_laravel
```

- e. Masuk folder project laravel, misal latihan 1

```
D:\project_laravel>cd latihan1
```

- f. Kemudian setelah masuk ke project laravel yang akan dicek versinya ketik perintah

```
D:\project_laravel\latihan1>php artisan --version
```

- g. Selanjutnya perhatikan output dari perintah diatas, misal dilayar tampak seperti berikut :

```
Laravel Framework 6.9.0
```

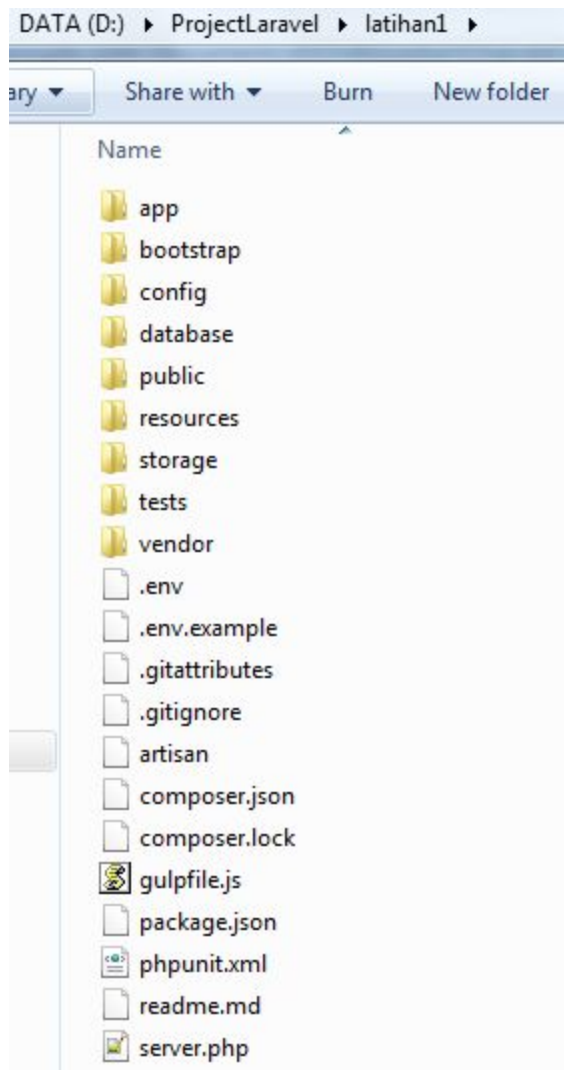
Artinya project laravel **latihan1** tersebut menggunakan versi 6.9.0

## F. Struktur Direktori Framework Laravel

Setelah proses instalasi maka akan terbentuk direktori baru sesuai dengan yang telah didefinisikan pada composer ketika memberikan perintah **create-project**, sebagai contoh pada kasus diatas akan terbentuk direktori baru yaitu **latihan1**.

Mengapa hal ini harus dibahas? laravel 5 memiliki susunan direktori yang berbeda dari framework-framework PHP lainnya apalagi bagi programmer yang move on dari

prosedural ke framework PHP. Secara garis besar, kebanyakan framework PHP yang menganut pola MVC (Model-View-Controller) menggunakan skema direktori dengan



nama "Model", "View" dan "Controller" yang seluruhnya dikumpulkan kedalam sebuah direktori utama yang bernama "src" atau "app" / "application" (seperti CodeIgniter). Pada skema tersebut, direktori "Model" digunakan untuk menyimpan class PHP yang berhubungan dengan model database, kemudian direktori "Controller" digunakan untuk menyimpan class PHP yang berhubungan dengan application logic dan direktori "View" digunakan untuk menyimpan file-file yang berhubungan tampilan aplikasi. Konvensi struktur direktori tersebut juga digunakan oleh laravel versi 5 namun dengan struktur yang sedikit berbeda dari biasanya dan (menurut saya) hal ini akan membuat bingung para pengguna framework yang baru berpindah dari framework lain seperti Yii / CodeIgniter. Pada gambar disamping adalah struktur

file dan direktori laravel setelah anda download menggunakan composer !

Berikut adalah struktur direktori laravel versi 5

### 1. app/Http

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

- a. **app/Http/Controllers** : Direktori ini digunakan untuk menyimpan seluruh class Controller yang kita buat seperti misalnya `ProductController.php`, `SalesController.php`, dll.
- b. **app/Http/Middleware** : Direktori ini digunakan untuk menyimpan seluruh class yang berhubungan dengan 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. Untuk penjelasan lengkapnya tentang middleware bisa baca dokumentasinya laravel.
- c. **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 atau pun dengan menggunakan perintah **php artisan make:request**. Untuk penjelasan lebih lanjut terkait penggunaan validator pada laravel, bisa dibaca dokumentasinya laravel.

## 2. database/migrations

Direktori ini berisikan file-file migrations yang di generate 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. Untuk penjelasan lebih lanjut terkait fitur migration pada laravel silahkan baca dokumentasinya laravel.

## 3. database/seeds

Direktori ini berisikan file-file database 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. untuk penjelasan lebih lanjut terkait fitur seeding pada laravel silahkan baca dokumentasinya laravel.

#### 4. 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 framework 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.

#### 5. Resources

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

- a. **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, sass 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. Untuk penjelasan lebih lanjut terkait fitur laravel elixir, silahkan baca dokumentasinya disini.
- b. **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. Untuk penjelasan lebih lanjut terkait penggunaan fitur localization pada laravel, silahkan baca dokumentasinya disini.
- c. **views** : Direktori ini digunakan untuk menyimpan seluruh file html / template blade yang kita buat.

#### 6. 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.

## G. Routing dan View

### a. Pengertian Routing

Setelah mempelajari bagaimana menginstal laravel tahap berikutnya adalah mempelajari routing. Routing merupakan hal utama ketika mempelajari sebuah framework, karena dengan mengetahui routing maka kita bisa mengatur jalannya program.

Route dalam bahasa indonesia berarti jalur, routing berarti mengatur jalur, jadi bisa kita ambil pengertian bahwa route pada laravel adalah bagian yang mengatur rute pada project aplikasi yang kita bangun dengan laravel. contoh kecilnya kita bisa mengatur rute url pada aplikasi yang kita buat dengan laravel. misalnya kita membuat route **pegawai** maka kita bisa memerintahkan untuk membuka **view**, menjalankan controller dan lain-lain pada saat route **pegawai** di akses.

Bagi anda yang sudah belajar framework Codeigniter tentu familiar dengan yang nama nya **View** dalam konsep MVC. Sama seperti pada framework PHP lainnya yang menggunakan konsep MVC, View bertugas untuk menangani atau mengurus bagian tampilan. Jadi segala sesuatu yang akan tampil pada user interface, akan kita buat pada view. misalnya menampilkan data dan lain-lain.

Tampilan awal project laravel ini berada pada view yang bernama **welcome.blade.php** letaknya ada di dalam folder **resources**. View **welcome.blade.php** tersebut diperintahkan secara **default** untuk tampil pada route **web.php** yang terletak dalam **folder routes**. Coba buka file **web.php** dalam folder routes, syntax seperti berikut :

```
<?php

/*
|-----
|
| Web Routes
|-----
|
|
```

```
| Here is where you can register web routes for
| your application. These
| routes are loaded by the RouteServiceProvider within a
| group which
| contains the "web" middleware group. Now create
| something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});
```

## b. Membuat Route Baru di Laravel

Pada saat framework laravel diinstal sudah membawa satu routes yaitu **web.php** yang terletak di folder **resources/views**. Adapun syntax dasar untuk membuat route adalah sebagai berikut :

```
Route::get(path, function ($param1,...,$param_n) {
    //statement
});
```

Sebagai latihan mari kita membuat route

### Latihan 1

buat route baru dengan nama **hello** yang berfungsi menampilkan tulisan 'Selamat datang di laravel', maka cara membuat nya adalah sebagai berikut :

1. Buka file **web.php** di folder **routes/web**
2. Ketik script berikut :

```
Route::get('/hello', function () {
    Echo 'Selamat datang di laravel';
});
```

3. Jalankan command jalankan aplikasi dengan mengetik

```
php artisan server (tekan enter)
```

4. Jalankan browser ketik url **http://127.0.0.1:8000/hello**





Gambar 5.15  
Hasil Route hello

## **Latihan 2**

Buat route baru dengan nama **DataPegawai** yang berfungsi menampilkan tabel data pegawai, maka cara membuat nya adalah sebagai berikut :

1. Jalankan editor (visual studio code / notepad++)
2. Buka file **web.php** di folder **routes/web**
3. Ketik script berikut :

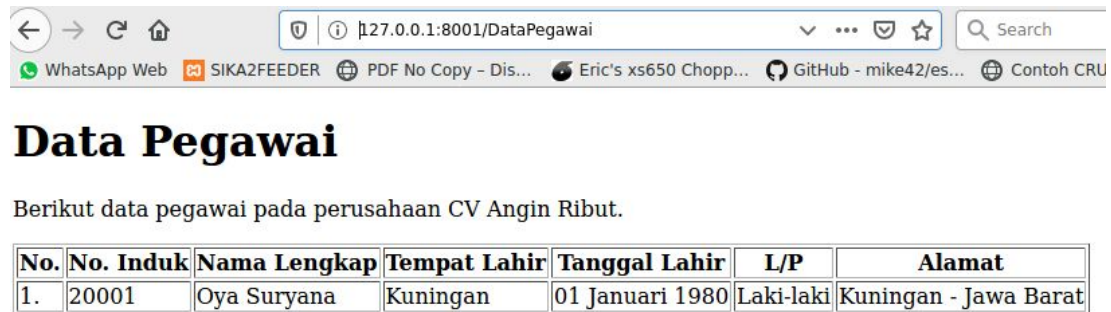
```
Route::get('/DataPegawai',function() {
    $KodeHtml  ='<!DOCTYPE html>';
    $KodeHtml  .='<html>';
    $KodeHtml  .='<head>';
    $KodeHtml  .='<title>Data Pegawai</title>';
    $KodeHtml  .='</head>';
    $KodeHtml  .='<body>';
    $KodeHtml  .='<h1>Data Pegawai</h1>';
    $KodeHtml  .='<p>Berikut data pegawai pada perusahaan CV
Angin Ribut.</p>';
    $KodeHtml  .='<table border="1">';
    $KodeHtml  .='<thead>';
    $KodeHtml  .='<tr>
        <th>No.</th>
        <th>No. Induk</th>
        <th>Nama Lengkap</th>
        <th>Tempat Lahir</th>
        <th>Tanggal Lahir</th>
        <th>L/P</th>
        <th>Alamat</th>
    </tr>';
    $KodeHtml  .='</thead>';
    $KodeHtml  .='<tbody>';
    $KodeHtml  .='<tr>
        <td>1.</td>
```

```

        <td>20001</td>
        <td>Oya Suryana</td>
        <td>Kuningan</td>
        <td>01 Januari 1980</td>
        <td>Laki-laki</td>
        <td>Kuningan - Jawa Barat</td>
    </tr>';
$KodeHtml .= '</tbody>';
$KodeHtml .= '</table>';
$KodeHtml .= '</body>';
$KodeHtml .= '</html>';
echo $KodeHtml;
});

```

4. Jalankan browser ketik url **http://127.0.0.1:8000/DataPegawai**



*Gambar 5.16  
Hasil Route DataPegawai*

Berdasarkan dua contoh diatas dapat diambil kesimpulan bahwa ketika user mengetik alamat url maka selanjutnya laravel akan menjalankan route sesuai nama rute yang dibuat, kemudian route yang dijalankan akan menjalankan fungsi yang ada didalam route tersebut, sebagai contoh ketika user mengetik url **http://127.0.0.1:8001/DataPegawai**, maka laravel akan menjalankan route **DataPegawai** yang ada pada file web.php di folder **routes**, dan pada route **DataPegawai** tersebut terdapat function yang menjalankan serangkaian perintah untuk menampilkan data dengan perintah echo, pada akhirnya halaman web menampilkan data sesuai yang tercantum dalam route yang dipanggil.

### c. Membuat route untuk memanggil view

Perhatikan latihan 2 diatas, meskipun secara output benar tampilan berjalan sesuai harapan, tetapi **secara konsep adalah salah !**, mengapa ??? karena tidak sesuai dengan konsep arsitektur Model View Controller, untuk itu agar sesuai dengan konsep arsitektur perlu membuat sebuah file yang berfungsi menampilkan objek dalam hal ini adalah view. Untuk itu script diatas akan dirubah dimana isi dari route **DataPegawai** akan dibuat menjadi file tersendiri yaitu file view dengan nama file **DataPegawai.blade.php**, adapun langkah-langkahnya adalah :

#### Latihan 3

Memisahkan antara route dengan view dengan cara membuat file view dan memanggilnya di route, berikut langkah-langkah pengerjaan :

1. Edit file route **DataPegawai** di file **web.php** pada folder **routes** menjadi seperti berikut :

```
Route::get('/DataPegawai',function(){  
    return view('DaftarPegawai');  
});
```

2. Kemudian buat file baru dengan nama **DaftarPegawai.blade.php** dan simpan di folder **resources/views**
3. Ketik script berikut :

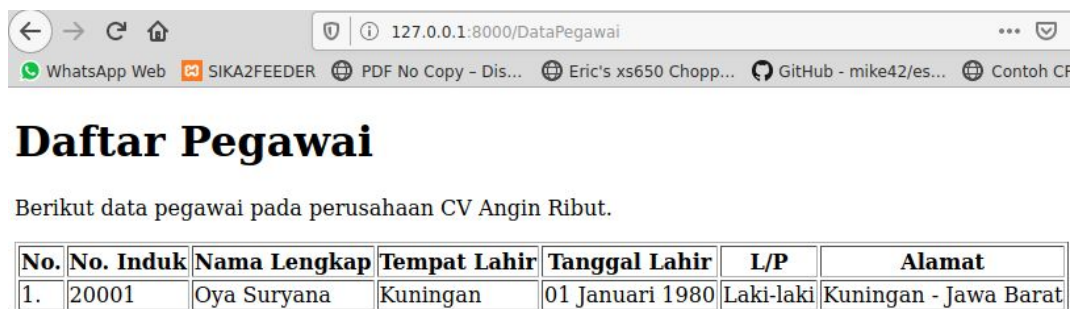
```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>Data Pegawai</title>  
    </head>  
    <body>  
        <h1>Daftar Pegawai</h1>  
        <p>Berikut data pegawai pada perusahaan CV Angin  
Ribut.</p>  
        <table border="1">  
            <thead>  
                <tr>  
                    <th>No.</th>
```

```

        <th>No. Induk</th>
        <th>Nama Lengkap</th>
        <th>Tempat Lahir</th>
        <th>Tanggal Lahir</th>
        <th>L/P</th>
        <th>Alamat</th>
    </tr>
</thead>
<tbody>
    <tr>
        <td>1.</td>
        <td>20001</td>
        <td>Oya Suryana</td>
        <td>Kuningan</td>
        <td>01 Januari 1980</td>
        <td>Laki-laki</td>
        <td>Kuningan - Jawa Barat</td>
    </tr>
</tbody>
</table>
</body>
</html>

```

4. Jalankan browser ketik url **http://127.0.0.1:8000/DataPegawai**



*Gambar 5.17  
Hasil Route DataPegawai*

Perhatikan hasil dari output latihan 3 diatas sama dengan latihan 2 ! apakah sama output layarnya ? apakah sama cara pembuatannya antara latihan 2 dengan latihan 3 ? jika berbeda silahkan analisis dimana letak perbedaannya !

#### d. Membuat route dengan parameter

Suatu saat ketika aplikasi semakin kompleks route yang kita buat diharuskan mampu mengirim data melalui URL (ingat dalam prosedural mengenai Variabel GET yang bersumber dari URL).

Bentuk URL yang dibuat dengan route dari latihan 1, 2 dan 3 adalah bentuk URL umum dengan syntax.

```
http://alamat_host/nama_route
```

Coba ingat kembali ke PHP prosedural, anda mungkin pernah membuat url address seperti berikut :

```
http://localhost/apl_pegawai/DetailPegawai.php?nik=20001
```

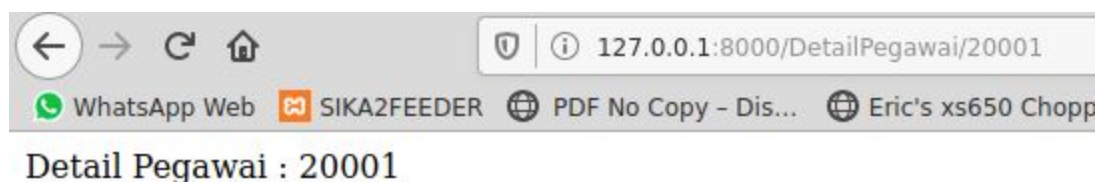
Perhatikan url diatas pada saat menjalankan file **DetailPegawai.php** url mengirim data ke file **DetailPegawai.php** dengan nama variabel **nik** dengan nilai **20001**, di Framework laravel kita tidak bisa melakukan pengiriman data melalui url dengan cara seperti itu, karena yang kita akses dalam laravel buka file php seperti dalam prosedural namun sebuah route.

Sebagai latihan mari kita buat sebuah route yang mengubah php prosedural diatas menjadi route dalam laravel. Ikuti langkah-langkah berikut :

1. Buka code editor anda, dan buka project laravel anda
2. Buka file **web.php** yang ada di folder **routes**
3. Tambahkan route baru dengan nama DetailPegawai dengan cara tambahkan script berikut diakhir baris file

```
Route::get('/DetailPegawai/{nik}',function($nik) {  
    echo 'Detail Pegawai : '.$nik;  
});
```

4. Jalankan browser ketik url **http://127.0.0.1:8000/DataPegawai/20001**



Gambar 5.17  
Hasil Route DataPegawai

Perhatikan pada route yang kita buat diatas, route nya mengirim data berupa serentetan angka **20001**, coba bandingkan dengan url pada php prosedural. Pada PHP prosuder untuk menangkap nilai didalam URL menggunakan variabel **GET**, sedangkan didalam framework Laravel membuat route yang mengandung parameter, nah nilai **20001** itulah yang disebut **parameter**, jadi route DetailPegawai tersebut memiliki **satu parameter** yaitu **nik**. Bagaimana sampai disini bisa membedakan cara mengirim data via URL di PHP prosedural dengan PHP Framework Laravel ?

Sekarang mari kita coba kalau parameter yang tertera dialam url dihapus sehingga anda mengakses alamat url diatas menjadi :

**`http://127.0.0.1:8000/DataPegawai`**

Apa yang terjadi ketika diakses oleh browser ? berikut adalah tampilan browser ketika kita mengakses route yang mengharuskan menggunakan parameter, namun parameternya tidak ada (dikosongkan) :



*Gambar 5.18  
Error karena mengosongkan parameter pada route*

Tampak pada gambar diatas terjadi error dengan kode error 404 artinya halaman web tidak ditemukan, jadi ketika membuat route dengan parameter jangan sekali-kali mengosongkan parameter yang diminta jika tidak akibatnya halaman aplikasi dianggap tidak ditemukan !

#### **e. Membuat route dengan optional parameter**

Bagaimana jika route yang kita buat bisa menggunakan parameter dan boleh tidak menggunakan parameter (bersifat opsional) kalau diisi maka parameter digunakan

kalau tidak diisi maka parameter diabaikan ? nah untuk kasus seperti ini maka kita gunakan parameter opsional artinya dibuat parameternya tetapi boleh diisi boleh tidak, sehingga menghindari error seperti tampak pada gambar 5.18 diatas.

Untuk memahami kasus ini mari kita buat kasus seperti ini, Buatlah sebuah route dengan nama **JumlahPegawai** untuk menampilkan data jumlah pegawai, parameternya adalah divisi, jika parameter diisi maka akan tampil jumlah karyawan untuk divisi yang disebutkan dalam parameter sedangkan jika parameter tidak diisi maka akan ditampilkan jumlah seluruh karyawan. Berikut adalah langkah membuat route tersebut :

1. Buka code editor anda, dan buka project laravel anda
2. Buka file **web.php** yang ada di folder routes
3. Tambahkan route baru dengan nama **JumlahPegawai** dengan cara tambahkan script berikut diakhir baris file :

```
Route::get('/JumlahPegawai/{divisi?}',function($divisi=null){
    if($divisi==null){
        echo 'Jumlah total pegawai : 1000 orang';
    } else {
        echo 'Jumlah pegawai divisi <b>'.$divisi.':</b> xxx
        orang';
    }
});
```

4. Jalankan browser ketik url **http://127.0.0.1:8000/JumlahPegawai/Programmer**



Gambar 5.19  
Tampilan route dengan parameter opsional

5. Percobaan berikutnya coba hilangkan parameter Programmer, sehingga url yang anda ketik adalah **http://127.0.0.1:8000/JumlahPegawai**, bagaimana hasilnya error kah ? ternyata tidak tapi menjalankan blok program yang parameter-nya null



Gambar 5.18  
Tampilan route saat parameter tidak diisi

#### f. Membuat route dengan regular expression (Regex)

Regular expression memungkinkan anda memformat parameter dengan aturan tertentu. Misal parameter hanya boleh diisi dengan angka, parameter hanya boleh diisi dengan huruf, atau parameter merupakan gabungan angka dan huruf membentuk pola tertentu, sebagai contoh kita ubah fungsi **DetailPegawai** dengan memformat parameter menggunakan regular expression, dengan ketentuan NIK hanya boleh diisi oleh angka, jika diisi dengan huruf aplikasi akan menampilkan error 404 (Not Found). Berikut langkah-langkah membuat parameter dengan regular expression :

1. Buka file **web.php** yang berada di folder **routes**

2. Edit bari route berikut :

```
Route::get('/DetailPegawai/{nik}', function($nik) {

    echo 'Detail Pegawai : '.$nik;

})->where('nik', '[0-9]+');
```

3. Menjadi seperti berikut :

```
Route::get('/DetailPegawai/{nik}', function($nik) {

    echo 'Detail Pegawai : '.$nik;

})->where('id', '[0-9]+');
```

4. Buka file **RouteServiceProvider.php** yang terletak di folder **app/Providers**, ubah baris berikut :

```
public function boot()
{
    //
    parent::boot();
}
```



5. Menjadi :

```
public function boot()
{
    //
    Route::pattern('nik', '[0-9]+');
    parent::boot();
}
```

6. Coba jalankan aplikasi dengan mengakses url

**http://localhost:8000/DetailPegawai/100**

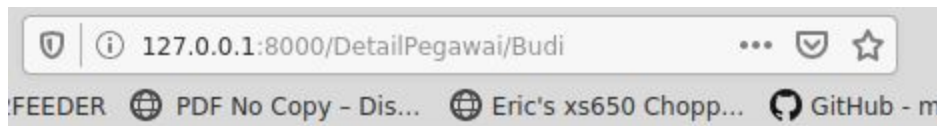


*Gambar 5.19  
Parameter diisi dengan angka*

7. Untuk membuktikan coba ganti parameter 100 diatas dengan huruf atau gabungan huruf dengan angka, misal Budi, P001 sehingga url diakses seperti berikut

**http://localhost:8000/DetailPegawai/Budi** atau

**http://localhost:8000/DetailPegawai/P001**



404 | Not Found

*Gambar 5.20  
Error karena parameter diisi dengan huruf*

Perhatikan hasilnya aplikasi menampilkan error 404, artinya ketika NIK diisi dengan karakter selain angka maka aplikasi akan menampilkan error, dimana akan dianggap bahwa halaman tidak ada

#### g. Membuat route untuk mengakses controller

Seluruh route yang dibuat diatas digunakan untuk mengakses view dengan perintah `return view('nama_view')` dan menampilkan dilayar dengan echo, namun sesungguhnya penggunaan route diatas tidak hanya untuk mengakses view saja namun akan sering digunakan untuk mengakses controller, berikut adalah contoh sebuah route dengan nama **DataPegawai** yang mengakses method **tampil()** di controller **PegawaiController**.

```
Route::get('/DataPegawai',PegawaiController@tampil);
```

### H. Controller

#### a. Pengertian Controller

Setelah memahami cara membuat route tahap selanjutnya kita akan mempelajari tentang controller, di dalam beberapa php framework file-file controller tersimpan di folder controller, termasuk di framework laravel. Laravel menyimpan controller di folder **app/Http/Controllers**.

Sama seperti controller pada framework PHP, karena sama-sama menerapkan konsep MVC dalam pengembangan dan penggunaan framework PHP nya. Controller merupakan jembatan atau penghubung antara view dan model. jadi secara mudah nya, controller bisa kita pahami sebagai pengatur view dan model. controller sendiri biasanya berperan sebagai pemberi perintah, dalam hal ini memerintahkan model untuk mengelola database dan memerintah view untuk menampilkan data sebagai output dari model. Mungkin di beberapa contoh kasus, kita memerlukan penerapan logika atau pengolahan data, maka controller lah yang berperan, baru kemudian ditampilkan ke bagian view (user interface) aplikasi atau project kita.

#### b. Membuat Controller

Untuk membuat file controller baru sangat mudah di laravel, dengan bantuan perintah baris dari php artisan kita bisa membuat controller, namun harus diperhatikan terutama tentang penamaan controller itu sendiri. Adapun cara untuk membuat controller ada dua cara yaitu :

1. Membuat secara manual file php baru disimpan di folder **app/Http/Controllers**
2. Membuat secara otomatis dengan perintah **php artisan**

Pada modul ini akan ditunjukkan cara pembuatan secara otomatis menggunakan php artisan saja. Penamaan Controller diawali dengan huruf besar jika nama file terdiri dari dua kata maka setiap awal kata diawali dengan huruf kapital misal **PegawaiController.php**, **SiswaController.php**, dsb.

Sebagai latihan mari kita buat controller Pegawai, adapun langkah-langkahnya adalah sebagai berikut :

1. Jalankan visual studio editor buka folder project **latihan1**
2. Jika tidak menggunakan visual studio editor jalankan command, masuk ke project (gunakan perintah CD jika menggunakan command)
3. Buat controller **PegawaiController** dengan perintah php artisan

```
D:\project_laravel\latihan1>php artisan make:controller  
PegawaiController (tekan enter)
```

4. Ketika perintah diatas selesai dieksekusi maka akan terbentuk file baru bernama **PegawaiController.php** di folder **app/Http/Controllers**
5. Selanjutnya anda buka file tersebut :

```
<?php  
  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
class PegawaiController extends Controller  
{  
    //  
}
```

6. Terlihat pada file tersebut sudah tersedia kerangka controller, tugas anda tinggal melanjutkan membuat method pada class blok class Controller.

### c. Membuat Method

Seperti telah kita pelajari di bab tentang OOP Native bahwa class merupakan kumpulan method (function), begitu pula controller, pada dasarnya controller adalah class turunan (*inheritance / extends*) dari kelas induk yaitu class **Controller**.

Controller di laravel merupakan kumpulan method, maka pada tahap selanjutnya adalah membuat Controller. Dalam satu controller minimal ada 1 method yang dibuat. Sebagai latihan mari kita buat sebuah aplikasi sederhana yaitu aplikasi kepegawian, dalam hal ini hanya akan membahas proses pembuatan route, controller dan view (web statis), tidak menggunakan database.

### **Langkah 1 (Membuat Route Home Untuk Tampil Data)**

Buat tampilan awal aplikasi ketika diakses home route akan menampilkan tabel data karyawan. Berikut langkah-langkahnya :

1. Buka file **web.php** yang tersimpan di folder **routes**
2. Edit baris berikut :

```
Route::get('/', function () {  
    return view('welcome');  
});
```

3. Ubah menjadi berikut :

```
Route::get('/', 'PegawaiController@tampilData');
```

Baris diatas artinya pada saat halaman home web (/) diakses akan menjalankan method **tampilData** yang ada di Controller **PegawaiController**

4. Selanjutnya buka file **PegawaiController.php** di folder **app/Http/Controller**, kemudian block function berikut (bagian yang di tulis tebal)

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
  
class PegawaiController extends Controller  
{  
    public function TampilData(){
```

```

        return view('TampilPegawai');
    }
}

```

Pada Controller **PegawaiController** diatas ditambahkan method baru yaitu method **TampilData**, yang berfungsi menjalankan sebuah file view yang bernama **TampilPegawai.blade.php** yang terletak di folder **resources/views** , jika file belum ada maka halaman akan error seperti berikut :



5. Untuk memperbaiki error tersebut anda harus membuat file view dengan nama file **TampilPegawai.blade.php**, di folder **resources/views** dan ketik script berikut :

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Data Pegawai</title>
</head>
<body>
    <h2>Data Pegawai</h2>
    <p>Berikut ini data pegawai yang terdaftar dalam
database</p>

    <table border="1">
        <thead>
            <tr>
                <th>No</th>
                <th>No. Induk</th>
                <th>Nama Pegawai</th>
                <th>L/P</th>
                <th>Nomor HP</th>
            </tr>

```

```

        </thead>
        <tbody>
            <tr>
                <td>1.</td>
                <td>19001</td>
                <td>Oya Suryana</td>
                <td>L/P</td>
                <td>085224191648</td>
            </tr>
        </tbody>
    </table>
</body>
</html>

```

6. Kemudian akses url **http://localhost:8000/**, jika tidak ada masalah maka akan tampil seperti berikut :



## Data Pegawai

Berikut ini data pegawai yang terdaftar dalam database

No	No. Induk	Nama Pegawai	L/P	Nomor HP
1.	19001	Oya Suryana	L/P	085224191648

### Langkah 2 (Membuat Route TambahData Untuk Menampilkan Form Tambah Data)

Dari tampilan diatas akan kita modifikasi agar terdapat hyperlink Tambah Data Pegawai diatas tabel yang berfungsi menjalankan route **TambahData** , dimana route tersebut akan menjalankan method **TambahData** pada controller **PegawaiController** dan method TambahData akan menjalankan view **TambahPegawai.blade.php**, berikut adalah tahapan untuk mengerjakan hal tersebut :

1. Buka file view **TampilPegawai.blade.php**, di folder **resources/views** dan lakukan pada script tersebut dengan menambahkan script yang dicetak tebal dibawah ini :

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Data Pegawai</title>
</head>
<body>
    <h2>Data Pegawai</h2>
    <p>Berikut ini data pegawai yang terdaftar dalam
database</p>

    <p><a href="TambahData">Tambah Data Pegawai</a></p>
    <table border="1">
        <thead>
            <tr>
                <th>No</th>
                <th>No. Induk</th>
                <th>Nama Pegawai</th>
                <th>L/P</th>
                <th>Nomor HP</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1.</td>
                <td>19001</td>
                <td>Oya Suryana</td>
                <td>L/P</td>
                <td>085224191648</td>
            </tr>
        </tbody>
    </table>
</body>
</html>

```

2. Kemudian akses url **http://localhost:8000/**, maka akan tampil perubahan dengan adanya link tambah diatas tabel :



## Data Pegawai

Berikut ini data pegawai yang terdaftar dalam database

[Tambah Data Pegawai](#)

No	No. Induk	Nama Pegawai	L/P	Nomor HP
1.	19001	Oya Suryana	L/P	085224191648

3. Selanjutnya tambahkan route baru yaitu **TambahData** yang berfungsi menjalankan method **TambahData** di controller **PegawaiController**

```
Route::get('/TambahData','PegawaiController@TambahData');
```

4. Kemudian edit controller **PegawaiController** tambahkan method baru yaitu **TambahData** (cetak tebal) yang menjalankan perintah untuk me-load view dengan nama **TambahPegawai.blade.php**, sehingga **PegawaiController** menjadi seperti berikut :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PegawaiController extends Controller
{
    public function TampilData(){
        return view('TampilPegawai');
    }

    public function TambahData(){
        return view('TambahPegawai');
    }
}
```



5. Berdasarkan method baru yang meminta untuk menjalankan view `TambahPegawai`, maka langkah berikutnya buatlah sebuah view dengan nama file `TambahPegawa.blade.php` dan simpan dalam **folder resources/views**, dan ketik script berikut :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Data Pegawai</title>
</head>
<body>
    <h2>Data Pegawai</h2>
    <p>Berikut ini data pegawai yang terdaftar dalam
database</p>
    <form method="POST" action="SimpanData">
        <p>Nomor Induk <br/>
            <input type="hidden" name="_token" value="<?php echo
csrf_token();?>"/>
            <input type="text" name="txtNoInduk"
placeholder="Masukan Nomor Induk"/>
        </p>

        <p>Nama Lengkap <br/>
            <input type="text" name="txtNamaLengkap"
placeholder="Masukan Nama Lengkap"/>
        </p>

        <p>Jenis Kelamin <br/>
            <input type="radio" name="radioJK"
value="L"/>Laki-laki
            <input type="radio" name="radioJK"
value="P"/>Perempuan
        </p>

        <p>Nomor Handphone <br/>
            <input type="text" name="txtNoHp"
placeholder="Masukan Nomor Handphone"/>
        </p>

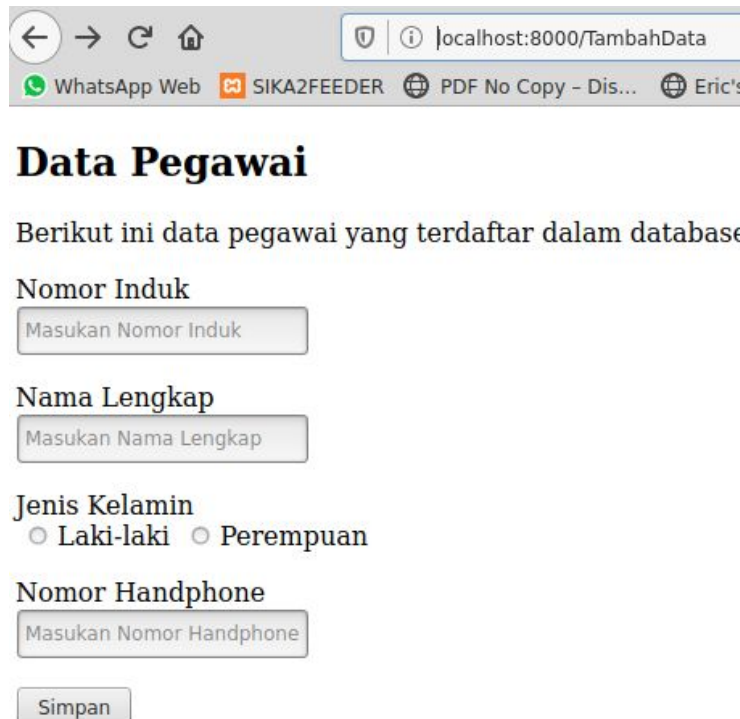
        <p>
```

```

        <button type="submit"
name="btnSimpan">Simpan</button>
    </p>
</form>
</body>
</html>

```

- Perhatikan pada script view **TambahPegawai.blade.php** diatas pada bagian action form disana tidak mengakses file php seperti di prosedural namun mengakses route yaitu route **SimpanData**, dan perhatikan pula terdapat komponen input yang bertipe hidden dengan nama **\_token** yang bernilai acak sebagai output dari fungsi `csrf_token()` laravel sebagai pengaman form dari serangan **Cross Site Request Forgery**.
- Untuk mengujinya coba anda klik hyperlink **Tambah Data Pegawai**, atau akses URL **http://127.0.0.1:8000/TambahData** jika menggunakan php artisan, jika menggunakan xampp **http://localhost/latihan1/TambahData**, jika berhasil maka akan tampak form berikut :



**Data Pegawai**

Berikut ini data pegawai yang terdaftar dalam database

**Nomor Induk**

**Nama Lengkap**

**Jenis Kelamin**  
☐ Laki-laki ☐ Perempuan

**Nomor Handphone**

### **Langkah 3 (Membuat Route SimpanData / Parsing Data Dari Form)**

Tahap berikutnya ketika user mengisi data karyawan baru yang terjadi adalah pengiriman data melalui form, pada bagian action yang dituju maka disanalah menangkap data / parsing data dari form, anda ingat ketika belajar prosedural proses parsing data dari form menggunakan variabel `$_POST['nama_komponen_input']`, jika sudah menggunakan framework laravel hal ini tidak berlaku lantas bagaimana mengambil data dari form input ? Ok berikut langkah-langkahnya\_:

1. Edit kembali file **web.php** yang ada di folder **routes**
2. Tambahkan route baru sesuai dengan yang disebutkan dalam form action yaitu **SimpanData** yang berfungsi menjalankan method **SimpanData()** di Controller **PegawaiController**, berikut adalah route yang harus ditambahkan dalam file **web.php** di baris mana saja dalam hal ini simpan saja di baris paling akhir :

```
Route::post('/SimpanData','PegawaiController@TambahData');
```

3. Perhatikan route diatas terdapat perbedaan yaitu anda menggunakan **Route::post** bukan **Route::get** lagi kenapa ? karena anda akan menangkap data dari Form !
4. Selanjutnya edit controller **PegawaiController.php** yang terletak di folder **app/Http/Controllers**, tambahkan method baru yaitu **SimpanData()**, perhatikan bagian script yang dicetak tebal, sehingga script **PegawaiController.php** menjadi seperti berikut :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PegawaiController extends Controller
{
    public function TampilData(){
        return view('TampilPegawai');
    }

    public function TambahData(){
        return view('TambahPegawai');
    }
}
```

```

    public function SimpanData(Request $request){
        $NoInduk = $request->input('txtNoInduk');
        $NamaLengkap = $request->input('txtNamaLengkap');
        $radioJK = $request->input('radioJK');
        $NoHp = $request->input('txtNoHp');

        echo '<p>Pegawai dengan data ' . $NoInduk . ' ' .
            $NamaLengkap . ' ' . $radioJK . ' ' . $NoHp . '
            Sudah disimpan !</p>';
    }
}

```

- Perhatikan method **SimpanData()** diatas, ada yang berbeda dengan method yang dijalankan oleh **Route::get()** perbedaan terletak pada function nya dimana method **SimpanData()** menerima input berupa **Request** yang disimpan dalam sebuah object dalam hal ini diberi nama **\$request**. Selanjutnya untuk memanggil property object tersebut cukup memanggil **\$namaobjek->input('name\_pada\_input\_form')** nya saja seperti contoh diatas untuk menerima data dari input **txtNoInduk** maka cukup memanggil **\$request->input('txtNoInduk')**.
- Sebagai bukti silahkan anda isi pada form diatas kemudian tekan tombol simpan dan hasilnya akan tampak seperti berikut :



### Langkah 3 (Membuat Route HapusData)

Selanjutnya kita membuat Route **HapusData** yang berfungsi menjalankan method **HapusData()** di Controller **PegawaiController**, adapun langkahnya adalah sebagai berikut :

- Buka kembali view **TampilPegawai.blade.php** yang terletak difolder **resources/views**,
- Edit file tersebut dengan menambahkan bagian yang dicetak tebal pada script dibawah, sehingga file view **TampilPegawai.blade.php** menjadi seperti berikut :

```
<!DOCTYPE html>
```

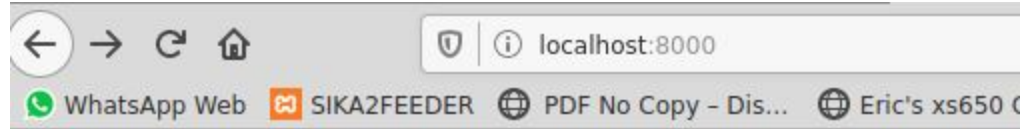
```

<html lang="en">
<head>
  <title>Data Pegawai</title>
</head>
<body>
  <h2>Data Pegawai</h2>
  <p>Berikut ini data pegawai yang terdaftar dalam
database</p>

  <p><a href="TambahData">Tambah Data Pegawai </a></p>
  <table border="1">
    <thead>
      <tr>
        <th>No</th>
        <th>No. Induk</th>
        <th>Nama Pegawai</th>
        <th>L/P</th>
        <th>Nomor HP</th>
        <th>Aksi</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1.</td>
        <td>19001</td>
        <td>Oya Suryana</td>
        <td>L/P</td>
        <td>085224191648</td>
        <td>
          <a href="HapusData/19001">Hapus</a>
        </td>
      </tr>
    </tbody>
  </table>
</body>
</html>

```

3. Jalankan kembali dengan mengakses URL **http://127.0.0.1:8000** atau dengan xampp di url **http://localhost/latihan1/public**, akan tampak perubahan dimana setiap baris data karyawan terdapat hyperlink Hapus seperti tampak pada gambar :



## Data Pegawai

Berikut ini data pegawai yang terdaftar dalam database

### [Tambah Data Pegawai](#)

No	No. Induk	Nama Pegawai	L/P	Nomor HP	Aksi
1.	19001	Oya Suryana	L/P	085224191648	<a href="#">Hapus</a>

- Setelah kita membuat href ke **HapusData**, maka selanjutnya kita membuat route baru dimana route yang kita buat membawa satu parameter yang tidak boleh dikosongkan (ingat kembali route dengan parameter), maka tambahkan baris berikut pada file **web.php** di folder **routes**

```
Route::get('/HapusData/{nis}','PegawaiController@HapusData');
```

- Selanjutnya edit controller **PegawaiController** dan tambahkan method baru **HapusData()** dengan satu parameter yaitu **nis**, sehingga file **PegawaiController** menjadi seperti berikut :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PegawaiController extends Controller
{
    public function TampilData(){
        return view('TampilPegawai');
    }

    public function TambahData(){
```

```

        return view('TambahPegawai');
    }

    public function SimpanData(Request $request){
        $NoInduk = $request->input('txtNoInduk');
        $NamaLengkap = $request->input('txtNamaLengkap');
        $radioJK = $request->input('radioJK');
        $NoHp = $request->input('txtNoHp');
        echo '<p>Pegawai dengan data '.$NoInduk.'
        '.$NamaLengkap.' '.$radioJK.' '.$NoHp.' Sudah disimpan
        !</p>';
    }

    public function HapusData($nik){
        echo 'Pegawai dengan '.$nik.' berhasil dihapus !';
    }
}

```

6. Selanjutnya anda coba klik hyperlink hapus, jika sukses akan tampil seperti berikut :



**Data Pegawai**

Berikut ini data pegawai yang terdaftar dalam database

[Tambah Data Pegawai](#)

No	No. Induk	Nama Pegawai	L/P	Nomor HP	Aksi
1.	19001	Oya Suryana	L/P	085224191648	<a href="#">Hapus</a>