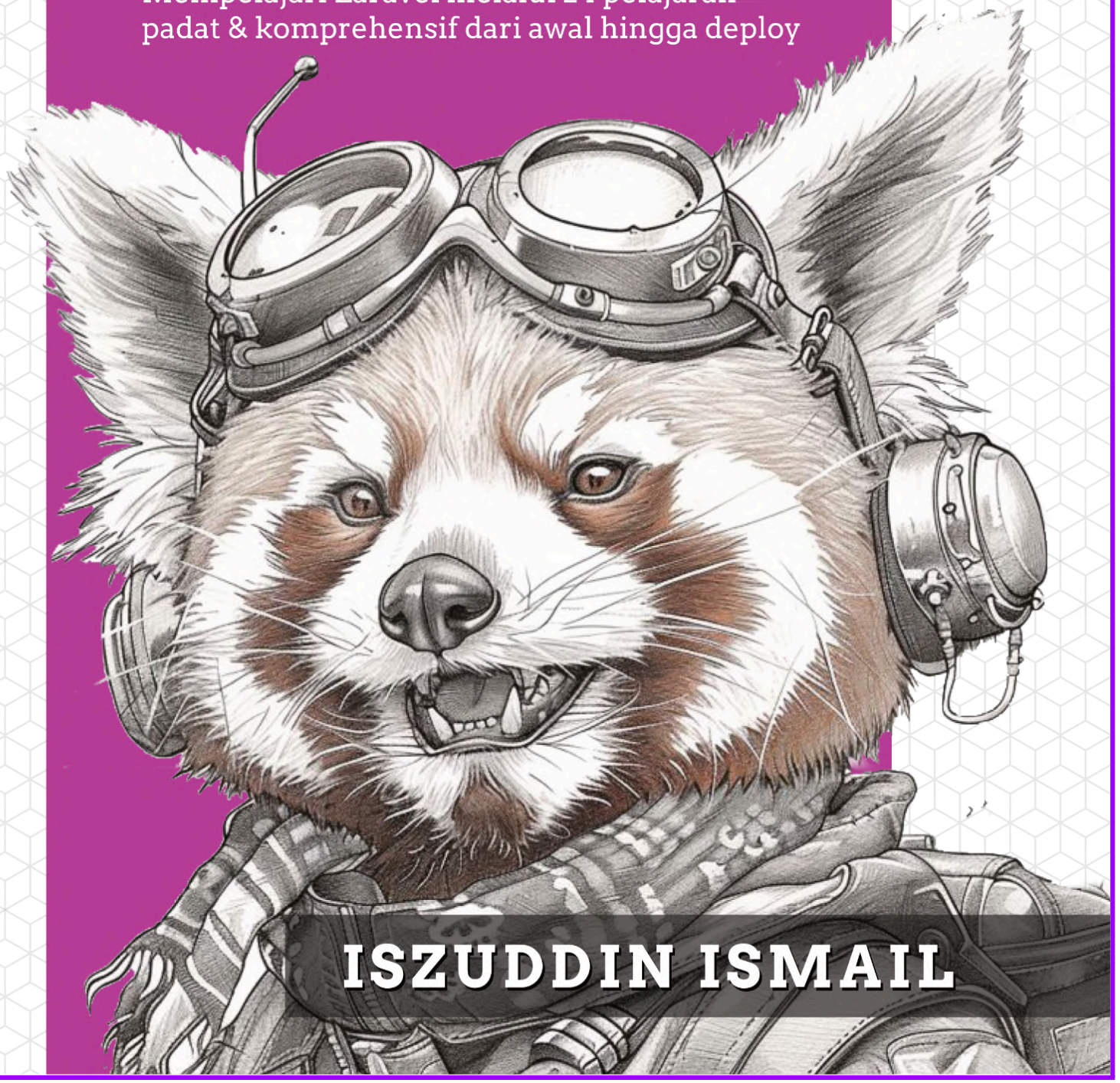
Sebuah projek **KELASPROGRAMMING.COM**

EDISI BAB CONTOH

24 Jam Laravel

Mempelajari Laravel melalui 24 pelajaran
padat & komprehensif dari awal hingga deploy



ISZUDDIN ISMAIL

Kandungan

Mengenai Buku Ini

Pastikan anda membuat pra-tempahan anda di :
🔗 https://kelasprogramming.com/24jl

Daripada penulis Iszuddin Ismail

Buku ini disusun untuk membantu mereka yang baru bermula dengan Laravel. Laravel ini memang cantik dan hebat. Ianya sangat memudahkan kerja-kerja web developer. Namun, bagi seseorang yang betul-betul baru, mungkin ianya sedikit mencabar untuk dipelajari.

Banyak topik dan pengetahuan baru yang diperlukan sebelum kita betul-betul boleh menggunakan Laravel dengan baik. Antaranya, termasuklah :

- Composer
- Autoloader
- Namespacing
- Arahan Artisan
- Konsep MVC
- Eloquent
- Blade
- Migration
- Seeder
- Tinker
- Dan lain-lain lagi

Sebab itulah buku ini diusahakan oleh saya dan pasukan di **KelasProgramming.com**. Kami berhasrat agar Laravel ini menjadi kemahiran lebih ramai *web developer* di Malaysia. Dan syarat dan kemahiran awal yang diperlukan tidak lagi menjadi halangan.

Melalui 24 bab dan 24 latihan, yang setiap satu boleh disiapkan dalam satu jam, saya berharap ianya akan memberikan permulaan yang baik kepada pemahaman anda terhadap Laravel. Seterusnya, anda akan mula membina kemahiran sebagai seorang web developer yang berkebolehan.

Buku ini akan disampaikan dalam bahasa Melayu dan dijangka akan berada dalam pasaran bermula **1 Januari 2025** pada harga **RM89.90**. Dianggarkan dengan semua topik yang akan disampaikan, ia akan mempunyai ketebalan **350 mukasurat**.

Pra-Tempahan

Untuk masa terhad, mereka yang berminat boleh membuat tempahan awal pada harga diskaun yang amat menarik.

Berikut adalah jadual pra-tempahan dan pelancaran buku 24 Jam Laravel.

1 Jun 2024	**Pra-tempahan dibuka** ***RM 40 + kos penghantaran & SST***
1 Ogos 2024	**Promosi Pra-pelancaran I** ***RM 50 + kos penghantaran & SST***
1 September 2024	**Suntingan & Semakan**
1 Oktober 2024	**Promosi Pra-Pelancaran II** ***RM 60 + kos penghantaran & SST*** **Rekabentuk & Rekaletak**
1 November 2024	**Cetakan**
1 Disember 2024	**Pelancaran** ***RM 70 + kos penghantaran & SST*** Fasa pelancaran dimulakan. Dalam tempoh ini buku dijual pada RM70 dan pembaca sepatutnya bakal menerima buku dalam tempoh 21 hari.
1 Januari 2025	**Pengedaran** ***RM 89.90 + kos penghantaran & SST*** Buku akan mula diedarkan kepada mereka yang telah membuat pra-tempahan. Buku juga akan mula dijual kepada pihak umum dengan harga jualan biasa. Kami akan mula menerima pembelian pukal dan juga promosi secara affiliate marketing.

Pengenalan

Bahasa & Istilah

Dalam menyampaikan pengetahuan pengaturcaraan, yang kebanyakan dimulakan dengan bahasa Inggeris, buku ini akan mengekalkan beberapa istilah asal dengan menggunakan ejaan bahasa Inggeris. Dengan ini, pembaca akan mudah membuat rujukan lanjut di dokumentasi rasmi projek Laravel, dan juga untuk pembelajaran lanjut dari sumber-sumber lain. Ini juga bagi mengelakkan kekeliruan dalam proses pembelajaran.

Berikut adalah istilah-istilah asal dari dokumentasi Laravel.

- *Router*
- *Controller*
- *View*
- *Model*
- *Event*
- *Queue*

Istilah sebegini kemudiannya akan ditulis dengan tulis senget seperti begini : *Router*, *Controller* dan *Event*.

Cara Buku Ini Disusun

Dalam sesuatu topik pembelajaran, terdapat banyak perkara yang perlu dibincang. Sekiranya kita merujuk kepada dokumentasi rasmi Laravel (https://laravel.com/docs) ia menyenaraikan kesemua ciri-ciri dan topik yang terdapat dalam Laravel. Namun dokumentasi tersebut, walaupun lengkap sesuai sebagai satu rujukan.

Dalam pembelajaran, saya berpendapat penuntut perlulah mempunyai konteks. Sebagaimana cara terbaik untuk mempelajari dan memahami fotosintesis dalam buku sains, selepas mempelajari teori, kita perlulah bercucuk tanam. Kita mula lihat bagaimana tanaman yang cukup sinar matahari, nutrien dari tanah, karbon dioksida akan membesar lebih hijau dan segar. Dan sekiranya sesuatu daripada elemen tersebut dipisahkan daripada tumbuhan, ia akan mengalami pembesaran yang terbantut.

Begitu juga dalam pembelajaran pengaturcaraan. Buku ini akan menyusun kandungan seperti ini :

- Bermula dengan masalah yang ingin diatasi
- Apa yang akan dibina untuk mengatasi masalah tersebut

- Apa yang digunapakai dalam membinanya

Sebagai contoh :

- Kita mahu memaparkan maklumat kepada pengguna
- Kita akan membina halaman dengan Laravel
- Kita akan gunakan
    - *Router* untuk menetapkan alamat URL dan
    - *View* untuk membina halaman dengan HTML

Di sini kita akan mempelajari bagaimana Router dan View digunakan.

Router dan *View* mempunyai banyak keadah dan cara digunakan. Jadinya, kita tidak akan mempelajari kesemuanya tentang *Router* dan *View* dalam satu bab. Bergantung kepada topik dan konteks yang sedang dibincangkan pada bab tersebut, *View* dan *Router* mungkin akan disentuh berkali-kali tetapi setiap kali, penulis akan memperkenalkan ciri-ciri dan keadah yang berbeza dalam menggunakannya.

Pra-Syarat

Saya mengandaikan bahawa pembaca mempunyai sedikit kemahiran dalam pengaturcaraan dan pembangunan web. Ini bermakna, pembaca mestilah mempunyai sedikit kemahiran dalam HTML, CSS, JavaScript dan PHP.

Sekiranya tidak, pembaca boleh mendapatkan akses ke Kursus Kilat PHP, HTML, CSS & JavaScript di https://kelasprogramming.com/register. Selepas mendaftar, pembaca akan mendapat akses ke Kursus Kilat HTML, CSS, PHP & JavaScript dalam bentuk rakaman video online.

Apa itu Laravel?

Saya mengandaikan pembaca mempunyai sedikit pengetahuan dalam pengaturcaraan dan juga PHP. PHP adalah bahasa pengaturcaraan yang menjadi pilihan ramai untuk membangunkan aplikasi web.

Dalam membangunkan aplikasi web, terdapat banyak fungsi-fungsi yang biasa digunakan seperti Authentication (proses *login*, *logout*, dsb), pemprosesan borang, membina rekabentuk dan lain-lain. Oleh sebab keperluan yang sama ini, walau apa pun aplikasi web yang dibina, mungkin sebuah himpunan fail-fail dengan fungsi-fungsi ini boleh disusun rapi untuk kegunaan berulang ini. Kita tidak perlu lagi menulis setiap daripada kosong setiap kali kita mahu membina sebuah aplikasi web.

Maka dapatlah kita sebuah "kerangka[1]" aplikasi web yang kemas dengan fungsi-fungsi asas yang diperlukan.

Dan kerangka ini ditambahbaik lagi dengan ciri-ciri keselamatan, fungsi khusus untuk pembangun, dan lain-lain.

Maka Laravel adalah kerangka aplikasi web untuk membantu pembangun membinanya dengan lebih baik, kemas, selamat dan tersusun.

Laravel telah mula dibangunkan oleh Taylor Otwell dari USA bermula 2011. Ia kini dimiliki oleh Laravel Holdings Inc., yang dikawalselia oleh Taylor Otwell sendiri. Laravel diedarkan dengan lesen sumber terbuka MIT, yang membenarkan semua aktiviti komersil.

Maklumat lanjut boleh didapati di https://laravel.com.

Dokumentasi Rasmi Sebagai Rujukan Utama

Buku ini ditulis untuk membantu warga Malaysia mula mempelajari dan menggunakan Laravel dalam membina aplikasi web. Apatah lagi apabila Laravel semakin menjadi pilihan baik dari sektor awam mahupun swasta.

Oleh demikian, saya ini memaklumkan bahawa adalah penting bagi pengamal Laravel sentiasa menjadi dokumentasi rasmi daripada Laravel sebagai rujukan yang paling utama.

Segala ciri-ciri dan maklumat tentang Laravel boleh didapati di https://laravel.com/docs.

Manfaat Menggunakan Kerangka?

Antara yang telah dibincangkan tadi, mana sahaja aplikasi web pasti mempunyai fungsi-fungsi yang sama seperti :

- Memaparkan maklumat dan halaman
- Authentication, iaitu proses login, logout dan seumpamanya
- Menerima dan memproses input pengguna
- Membaca data dari pengkalan data

Menggunakan kerangka aplikasi web seperti Laravel memberikan manfaat kepada dua pihak iaitu Pembangun dan Organisasi.

Antara manfaat menggunakan Kerangka seperti Laravel adalah :

[1] **Kerangka** - *Framework*. Seperti Laravel, himpunan fail pengaturcaraan dengan alatan, *library*, dengan fungsi-fungsi khusus menjadikannya sebuah asas kepada pembinaan aplikasi web.

- Kod sumber yang lebih kemas dan mudah diselenggara. Sistem yang dibangunkan pastinya akan ditambahbaik dari masa ke masa. Dengan sebuah kerangka yang telah menetapkan cara dan kaedah yang sama untuk semua pembangun, kod yang dibina menjadi lebih kemas dan tersusun. Ini memudahkan pengubahsuaian, pembaikan dan selenggara ke atas kod sumber sistem.

- Ahli pasukan baru dapat bermula dengan lebih pantas. Organisasi perlu menguruskan ahli pasukan pembangun yang bertambah atau berkurangan mengikut kepada situasi tertentu. Kerangka seperti Laravel menetapkan kaedah pembangunan yang khusus. Sebagai contoh, Laravel telah menetapkan di mana kod untuk pengkalan data perlu ditulis, di mana kod untuk HTML perlu disimpan dan sebagainya. Ditambah dengan dokumentasi rasmi yang lengkap dan terbuka, setiap pembangun baru boleh merujuk dokumentasi tersebut dan tidak memerlukan masa yang lama untuk memahami kod terdahulu.

- Mengelakkan kod berulang. Satu konsep yang sering digunapakai dalam pembangunan perisian adalah DRY atau *don't-repeat-yourself*. Konsep DRY ini menyatakan bahawa fungsi yang sama tidak sepatutnya ditulis di tempat yang lain. Tapi sepatutnya, kod yang sama diguna semula di setiap tempat ia diperlukan. Laravel membantu pembangun lebih peka terhadap konsep ini, menjadi kod sumber lebih efektif dan tidak serabut.

- Ciri-ciri keselamatan. Laravel telah dibangunkan dengan pelbagai ciri keselamatan yang terbina. Sebagai contoh, apabila menerima input pengguna, ia telah memeriksa input tersebut bebas dan selamat daripada isu keselamatan seperti *SQL injection*, CSRF (cross-site-request-forgery) dan lain-lain.

- Kerangka sumber terbuka mempunyai sokongan dan komuniti yang meluas. Terdapat banyak sumber rujukan daripada komuniti Laravel yang begitu besar di serata dunia. Rujukan baik dalam bentuk buku, video, tutorial, laman web dan pelbagai lagi terdapat di Internet.

- Pelbagai *library* dan pakej daripada komuniti. Laravel sendiri sudah menyokong pelbagai fungsi. Namun, terdapat juga ribuan lagi *library* dan pakej daripada komuniti untuk menyokong pelbagai fungsi lain. Baik untuk sebuah laman web *e-commerce*, *blog*, API, mahupun untuk sebuah halaman web ringkas, semuanya boleh dibangunkan dengan Laravel. Terdapat juga *library* bagi membantu membina fungsi-fungsi yang canggih seperti penggunaan Laravel di infrastruktur *serverless*, membina fungsi *multi-tenant*, dan lain-lain.

Dan pastinya banyak lagi manfaat menggunakan Kerangka aplikasi web seperti Laravel ini untuk pembangun mahupun organisasi.

Persiapan Komputer Pembangunan

Sama ada menggunakan laptop ataupun komputer, Windows mahupun Mac, ia perlu dipasang dengan beberapa aplikasi supaya kita boleh mula membina sebuah aplikasi.

Keperluan Minima Laravel

Untuk mula menggunakan Laravel, berikut adalah aplikasi minima yang diperlukan :

1. PHP (versi 8.2 untuk Laravel 11, juga diperlukan untuk Composer)
2. Node (diperlukan untuk aplikasi NPM kelak)
3. Git (diperlukan sekurang-kurangnya untuk *deployment* ke server kelak)

Untuk pengkalan data, biasanya kita menggunakan MySQL. Namun sebagai keperluan minima, kita boleh gunakan SQLite. SQLite adalah pengkalan data bersifat fail yang tidak memerlukan apa-apa aplikasi dipasang. Ia akan membina sebuah fail bersama dengan fail-fail Laravel nanti.

Untuk *web server*, biasanya kita menggunakan Nginx ataupun Apache. Dan ia diperlukan untuk *deployment* ke server di Internet kelak. Namun untuk pembangunan kita boleh menggunakan *web server* yang terbina bersama PHP bersama dengan arahan *CLI Artisan* kelak.

Laragon - Persiapan Untuk Windows

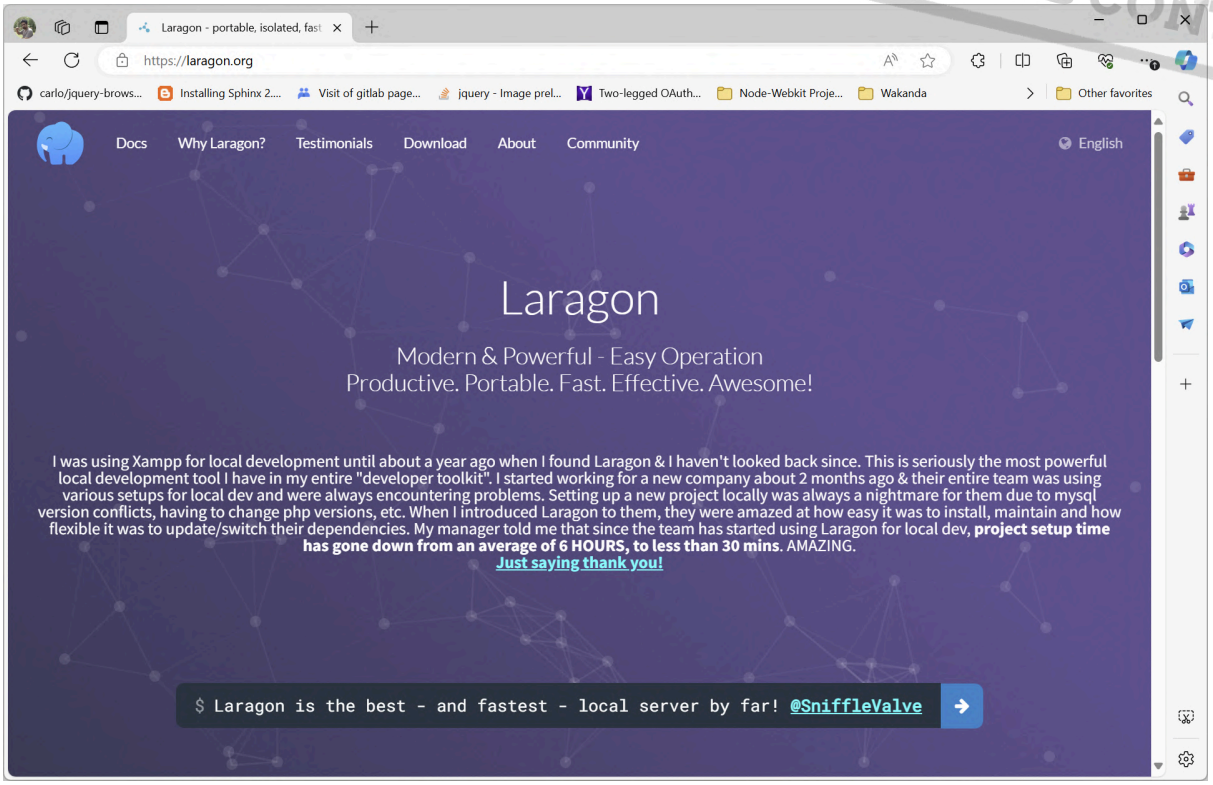
Walaupun keperluan minima hanya memerlukan 3 aplikasi, untuk pengguna Windows, buku ini mencadangkan Laragon untuk persiapan komputer pembangunan.

Kenapa Laragon dicadangkan :

- Didatangkan lengkap dengan pelbagai aplikasi yang diperlukan seperti pelbagai versi PHP, MySQL, web server Apache, pengkalan data MySQL, aplikasi klien HeidiSQL untuk membaca pengkalan data, Terminal untuk melaksanakan arahan CLI, Git, Node dan lain-lain.
- Tidak perlu memasang aplikasi berasingan
- Menyerupai persekitaran *server* Linux untuk produksi kelak
- Fungsi *virtual host* yang membantu penetapan alamat *URL* dan *folder* berbeza untuk setiap projek

Pastikan anda mendapatkan Laragon versi penuh (Full).

https://laragon.org/download/



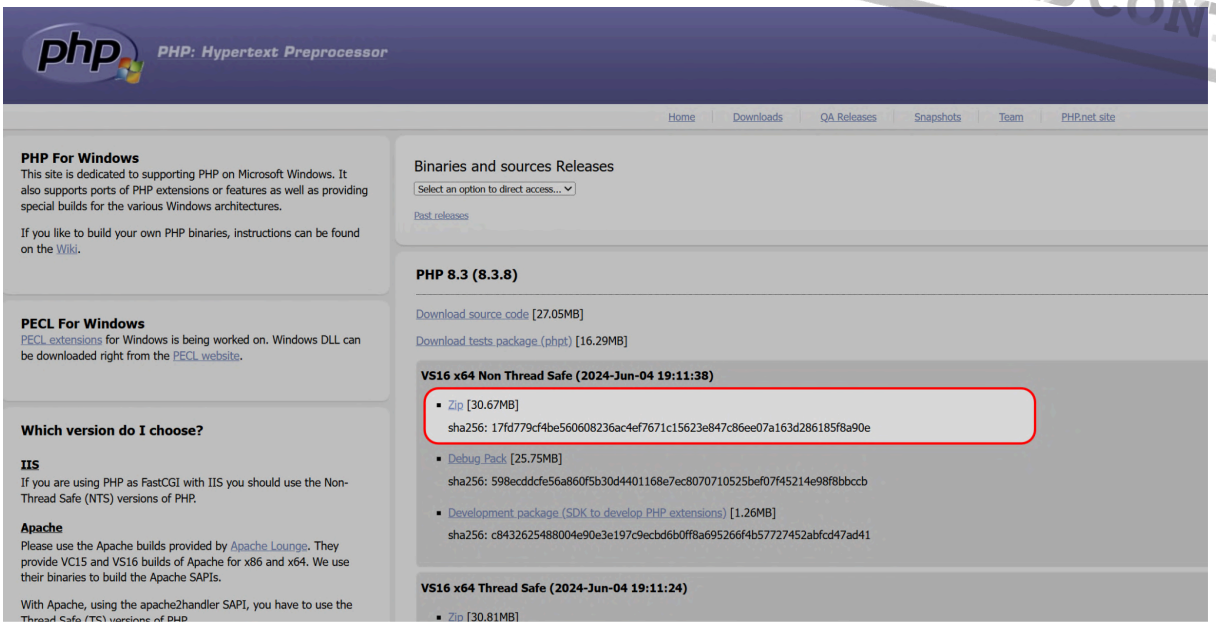
Pemasangan Versi PHP 8.2 / 8.3

Oleh kerana Laravel versi 11 memerlukan PHP versi 8.2 atau lebih, kita perlu mendapatkannya di halaman rasmi PHP. Untuk mendapatkan PHP versi 8.2, ia boleh dimuat-turun di :

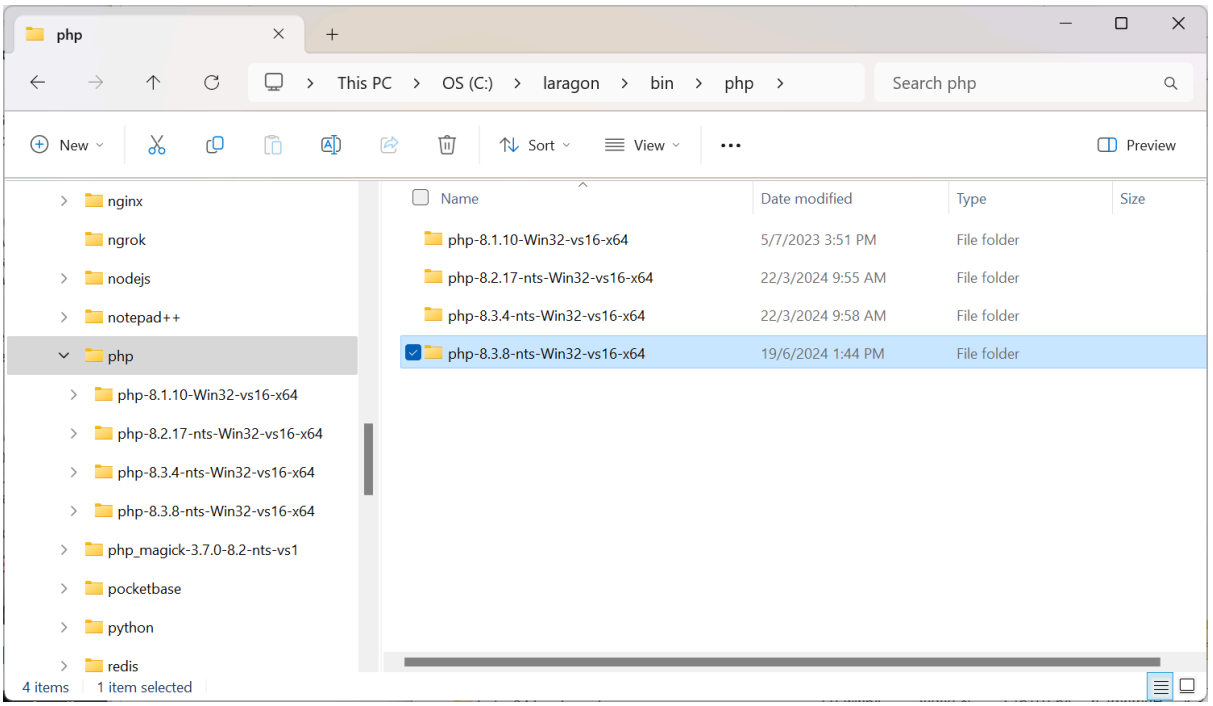
https://windows.php.net/download/

Pastikan anda dapatkan versi x64 *Non-Thread Safe* (NTS). Sebenarnya untuk *web server* Apache, kita boleh gunakan versi *Thread-Safe* ataupun *Non-Thread Safe*. Tetapi untuk *web server* Nginx, kita mesti gunakan *Non-Thread Safe*. Oleh sebab Laragon mempunyai kedua-dua Apache dan Nginx, adalah lebih baik kita gunakan versi yang boleh digunakan di kedua-duanya, iaitu *Non-Thread Safe*.

Muat-turun versi Zip.



Ekstrak fail Zip tersebut di folder C:\laragon\bin\php



Sesudah itu, sepatutnya pilihan versi PHP sudah boleh dilihat dalam Laragon.

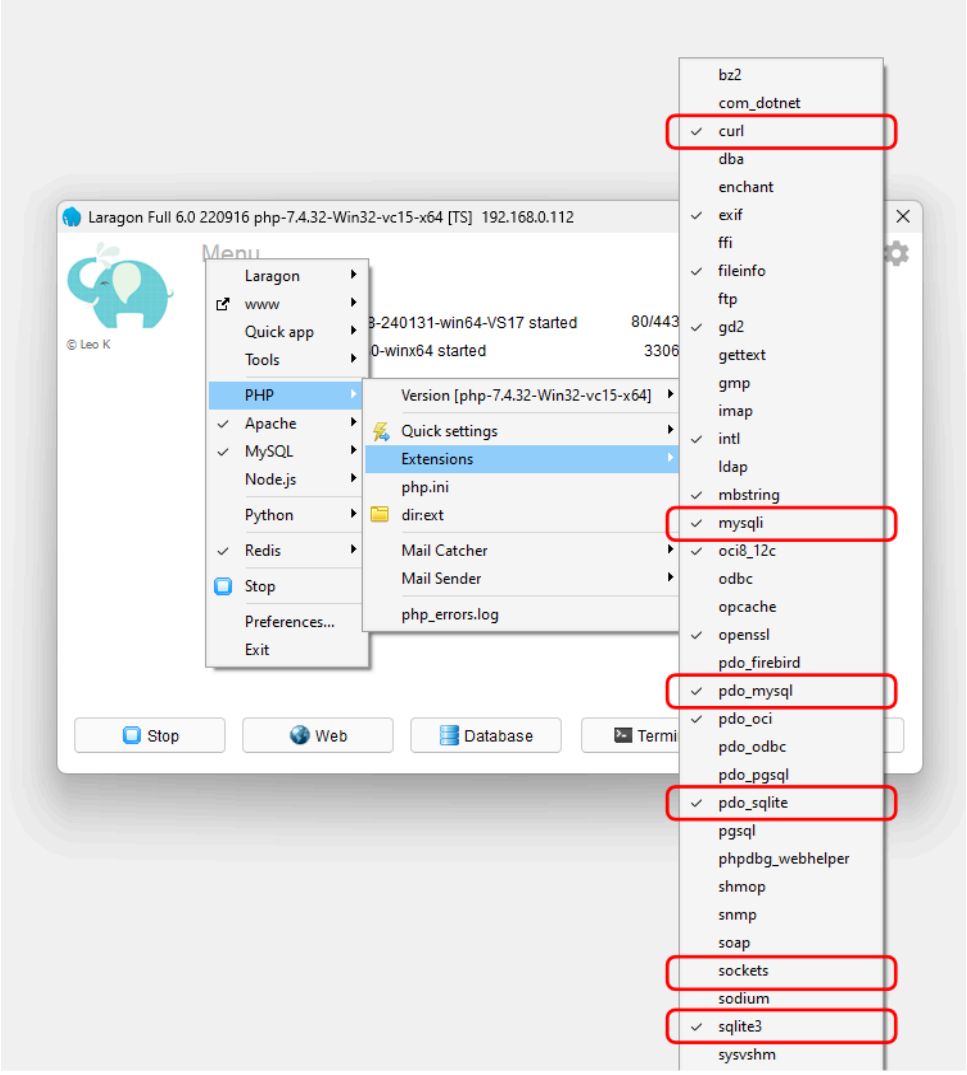
Mengaktifkan *Extension* PHP yang diperlukan

Setelah memasang versi PHP, kita perlu mengaktifkan beberapa extension PHP yang diperlukan dalam pengaturcaraan. Keupayaan sambungan pengkalan data untuk MySQL, SQLite dan sebagainya dibuat melalui *extension* ini. Maka beberapa tetapan perlu dilakukan untuk memastikan ia telah diaktifkan.

Dari Laragon, *right-click*[2] dan terus ke PHP > Extensions.

Berikut adalah beberapa *extension* yang biasa digunakan.

Di dalam situasi lain, sekiranya anda menghadapi ralat yang menyatakan fungsi tertentu tiada dalam PHP, mungkin *extension* yang diperlukan belum diaktifkan.

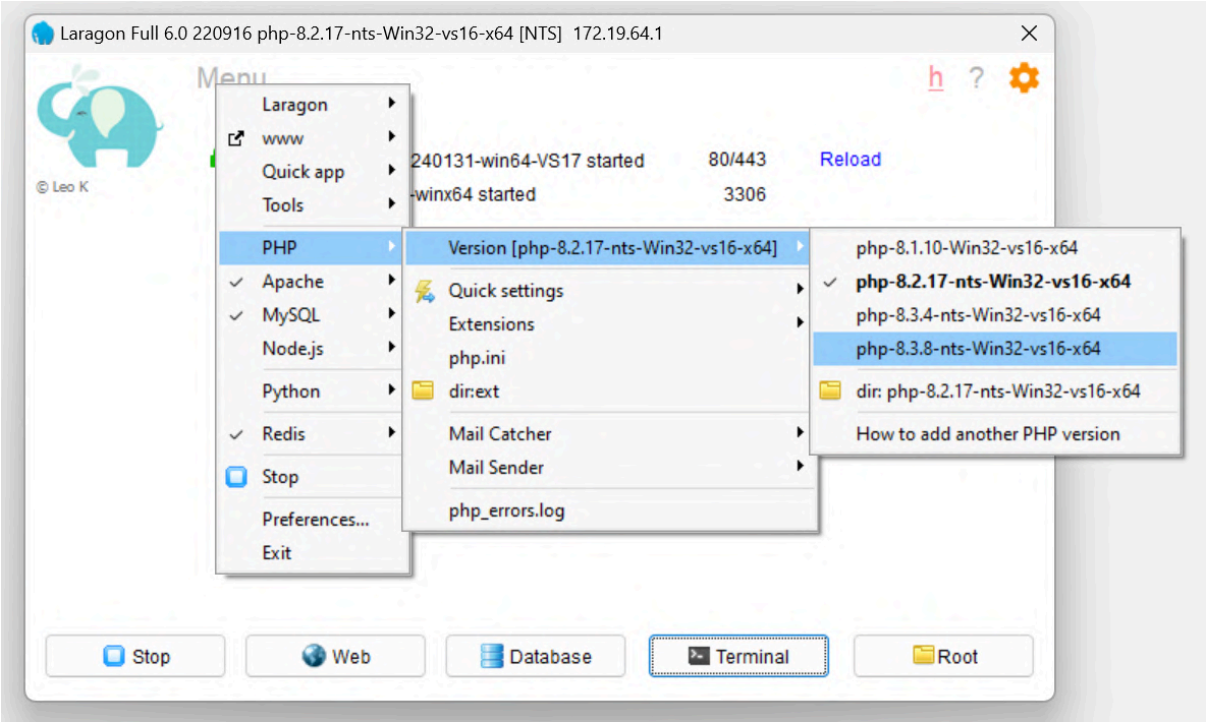


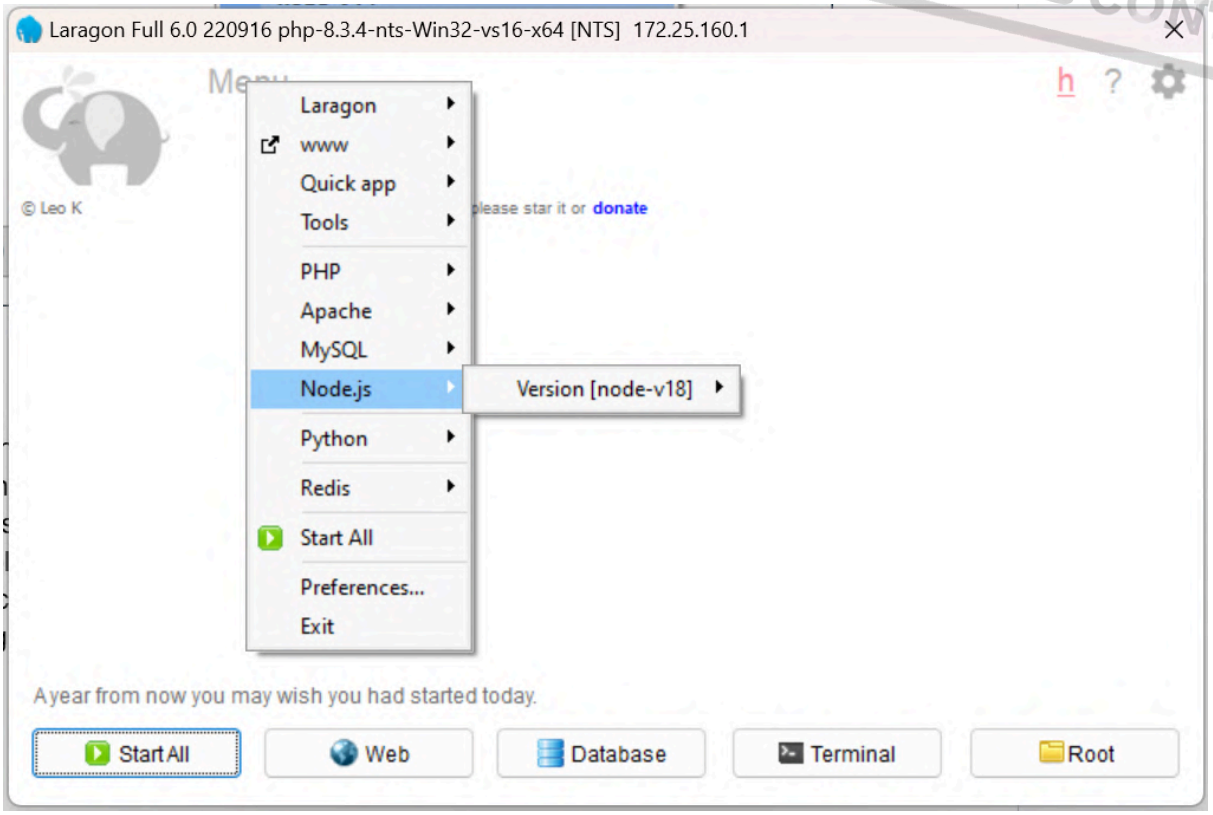
[2] Right-click - menekan butang kanan di tetikus

Kemaskini NPM / NodeJS

NodeJS dan NPM digunakan apabila kita mula menggunakan *library* JavaScript dan CSS seperti Tailwind dan Bootstrap CSS. Pastikan anda mengemaskini NodeJS ke versi yang terkini.

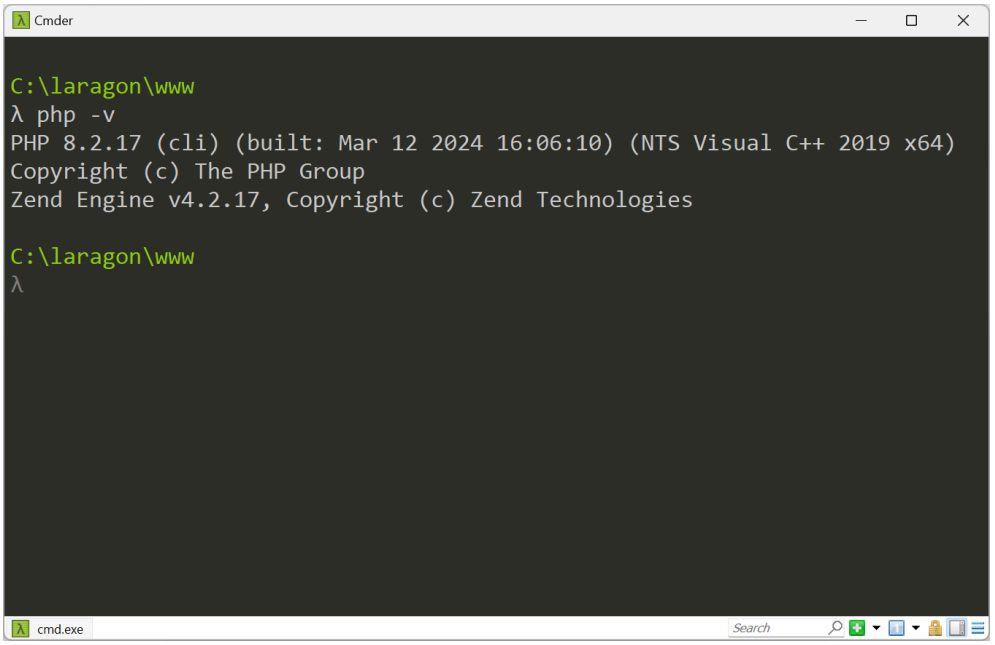
1. Dapatkan NodeJS terkini di https://nodejs.org/en/download/
2. Muat-turun versi LTS, Windows Binary.
3. Ekstrak fail ZIP ke folder C:\laragon\bin\nodejs (setiap versi di dalam folder sendiri)
4. Dari Laragon, *right-click* dan teruskan ke Node.js > Version.
5. Pilih versi Node.js yang telah dimuat-turun tadi.





Menggunakan Terminal

Apabila perlu menggunakan *Terminal* untuk arahan *NPM*, *Composer*, *Artisan* atau lain-lain kelak, buka *Terminal* melalui butang *Terminal* di Laragon.



Kemaskini Composer

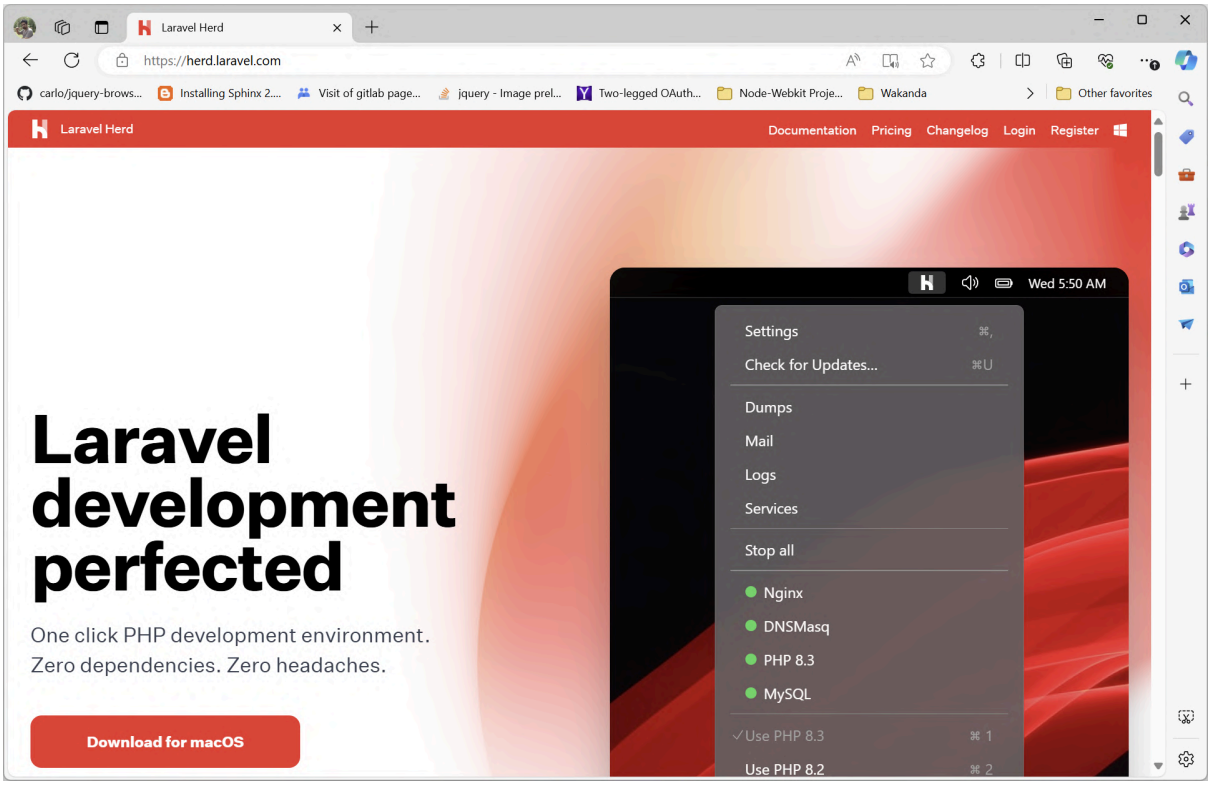
Sebelum pemasangan Laravel, sila kemaskini versi Composer. Buka Terminal daripada Laragon dan laksanakan arahan ini :

```
composer selfupdate
```

Laravel Herd - Persiapan untuk MacOS

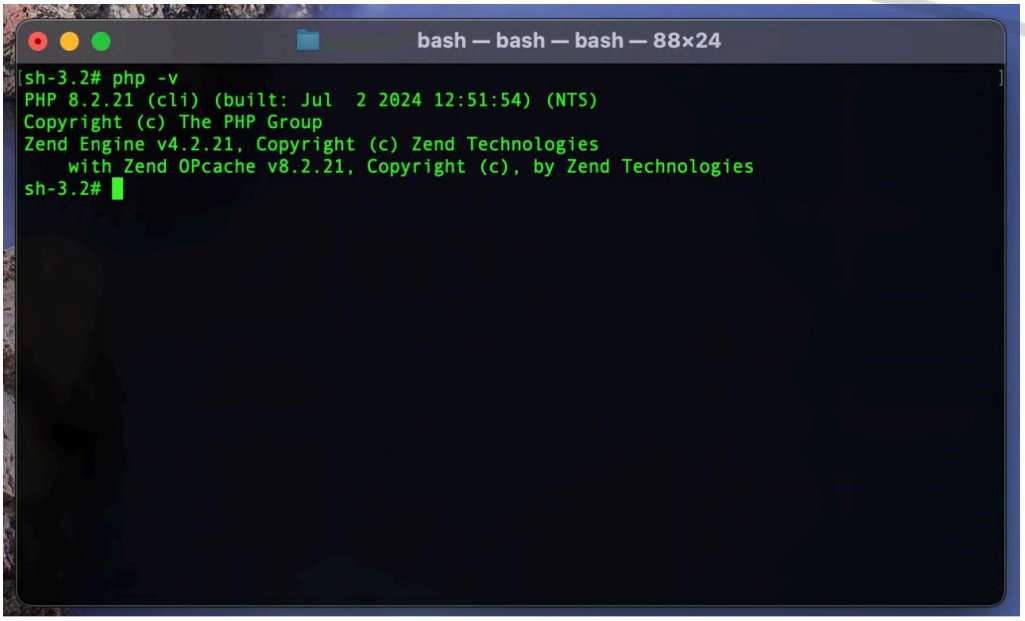
Walaupun terdapat Laravel Herd untuk Windows, saya masih mencadangkan Laragon untuk Windows. Namun untuk pengguna MacOS anda boleh gunakan Laravel Herd.

https://herd.laravel.com/



Menggunakan Terminal

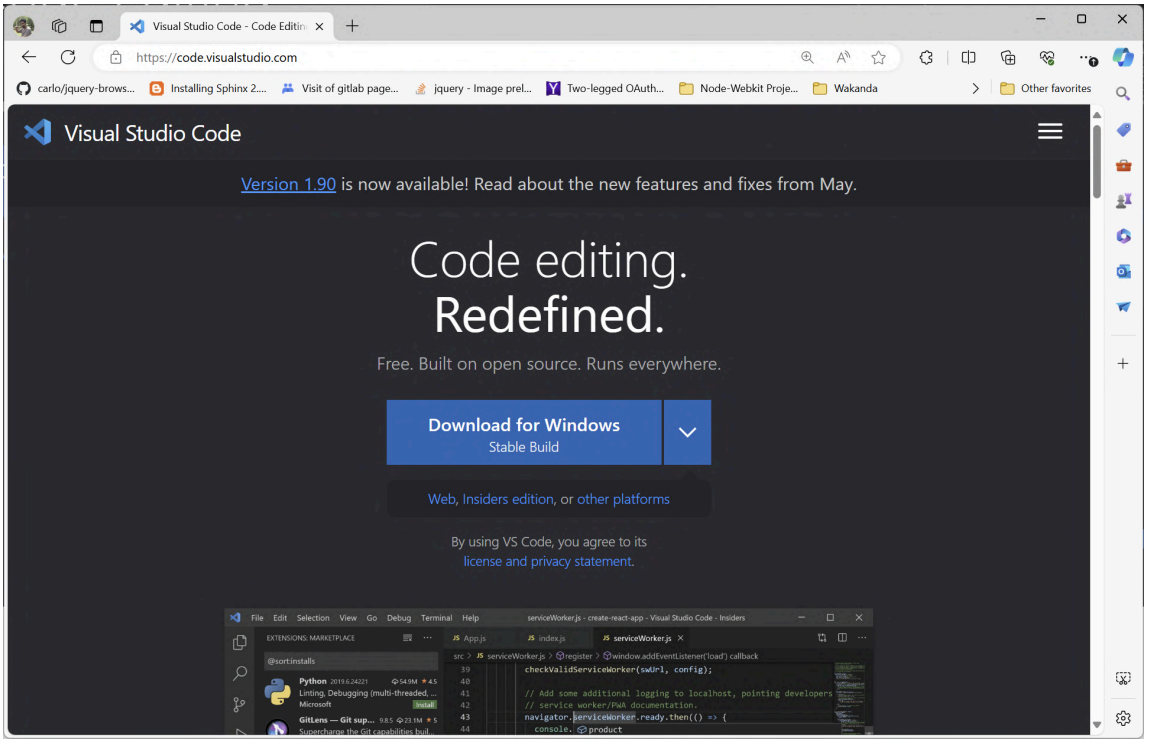
Di MacOS, Laravel Herd boleh digunakan bersama *terminal* sedia ada, iTerm dan Warp.



Code Editor Visual Studio Code

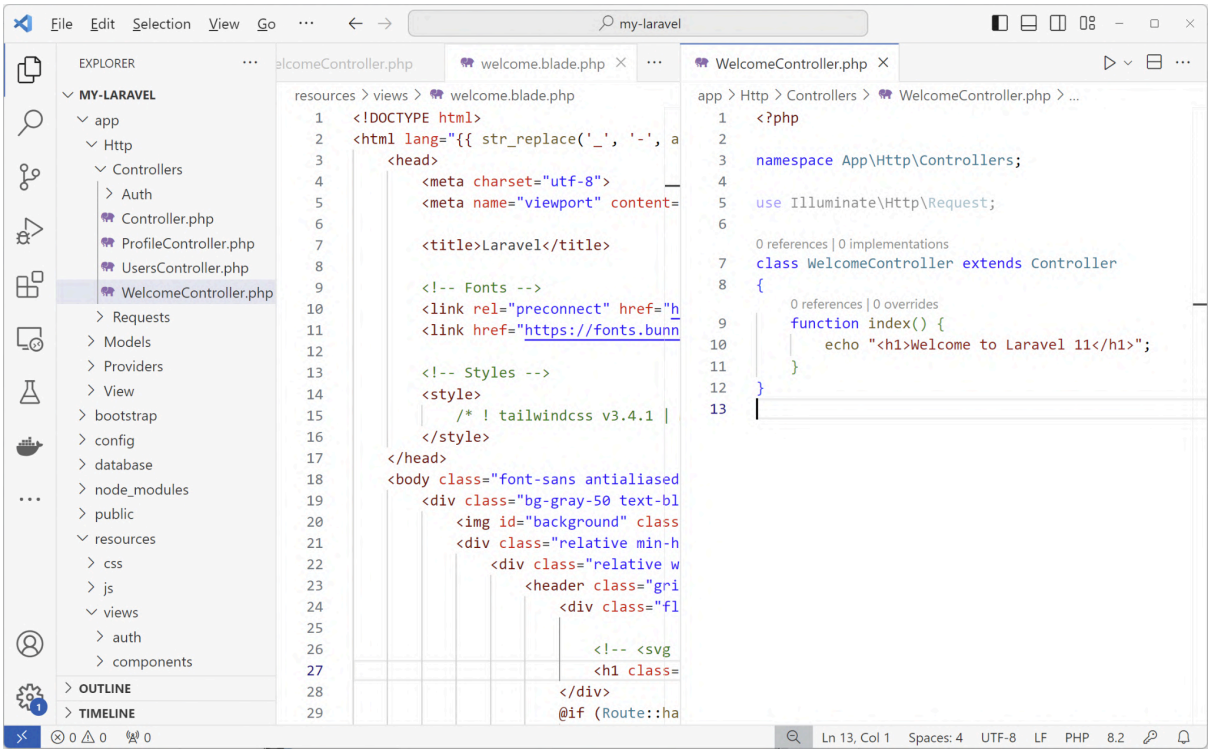
Untuk menulis kod, buku ini mengesyorkan Visual Studio Code. Visual Studio Code boleh didapati dari URL berikut :

https://code.visualstudio.com/



Visual Studio Code boleh digunakan bersama MacOS, Windows dan Linux. Muat-turun dan pasang Visual Studio Code ke komputer anda.

Extension VS Code Untuk Laravel

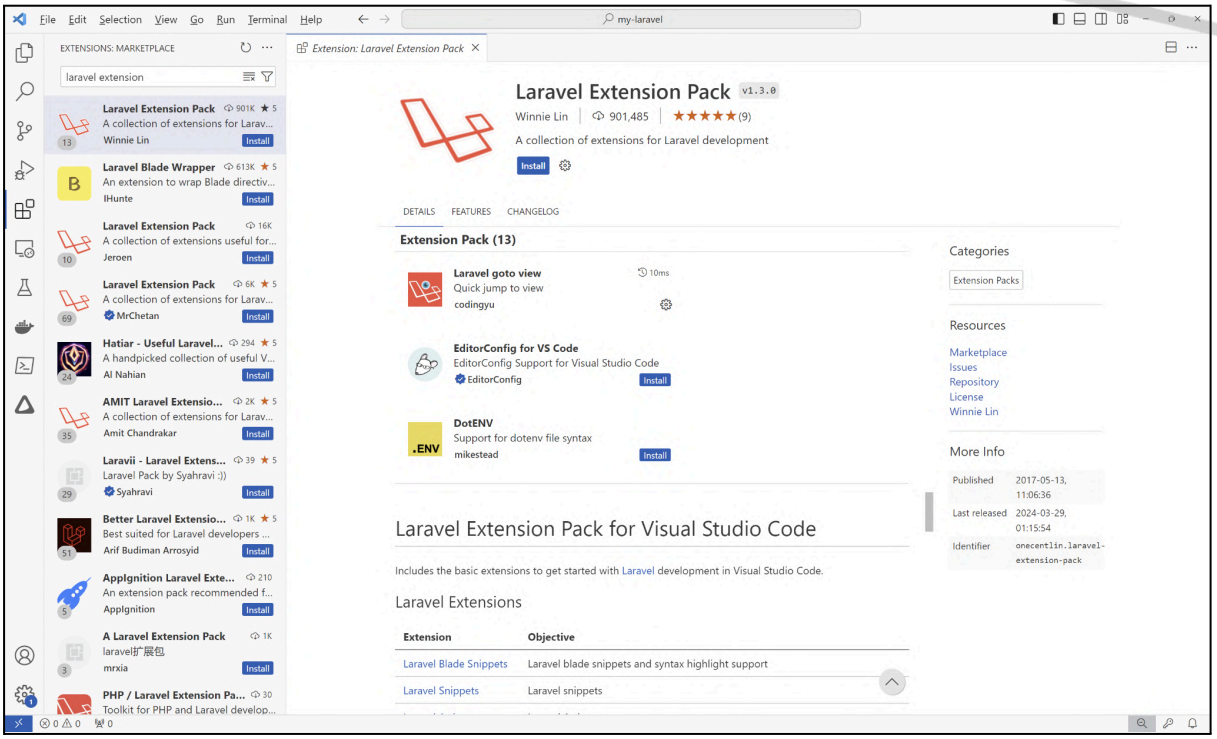


Dari ikon di sebelah kiri, pergi ke ikon bergambar kotak 

Buat carian dan dapatkan extensions berikut. Ada beberapa extension yang dicadangkan. Mungkin lebih mudah sekiranya kita memasang Laravel Extension Pack yang akan membantu memasang 12 extension untuk membantu pembangunan Laravel.

1. Laravel Blade Snippets
2. Laravel Snippets
3. Laravel Artisan
4. Laravel Extra Intellisense
5. Laravel goto view
6. laravel-jump-controller
7. laravel-goto-components
8. Laravel Blade formatter
9. Laravel Create View
10. Laravel Blade Wrapper
11. DotENV

12. DevDb



Membina Aplikasi Laravel

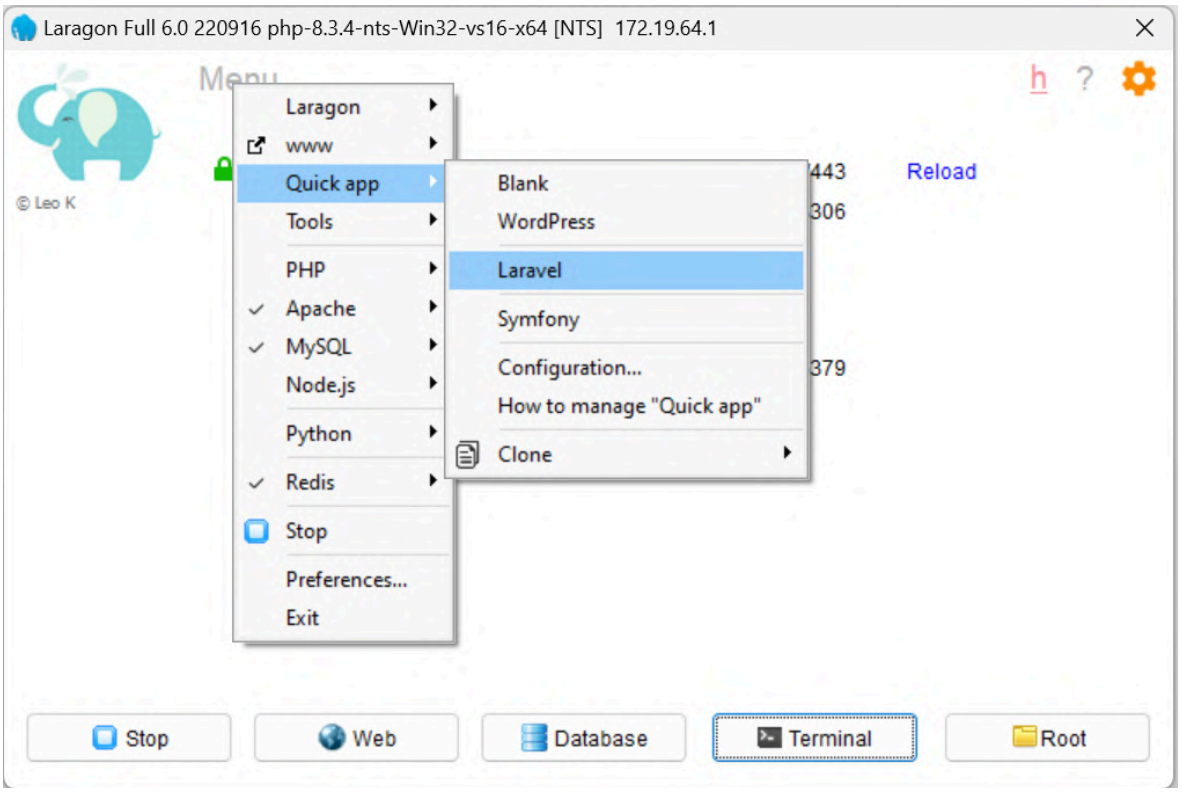
Laravel begitu fleksibel sehinggakan terdapat banyak cara untuk memulakan sebuah projek aplikasi web dengan Laravel. Berikut adalah beberapa caranya :

1. Menggunakan Composer
2. Menggunakan Laravel CLI
3. Laravel Sail dengan Docker

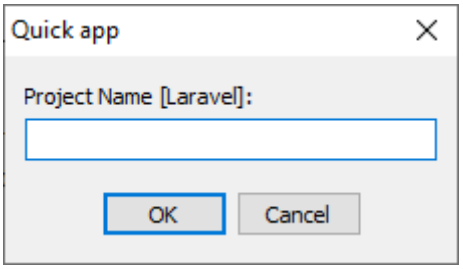
Namun untuk pembangunan, kedua-dua Laragon (Windows) dan Laravel Herd (MacOS) mempunyai ciri-ciri yang terbina untuk memudahkan membina aplikasi Laravel.

Aplikasi Laravel di Laragon (Windows)

Dengan Laragon, mulakan dengan Menu > Quick App > Laravel.



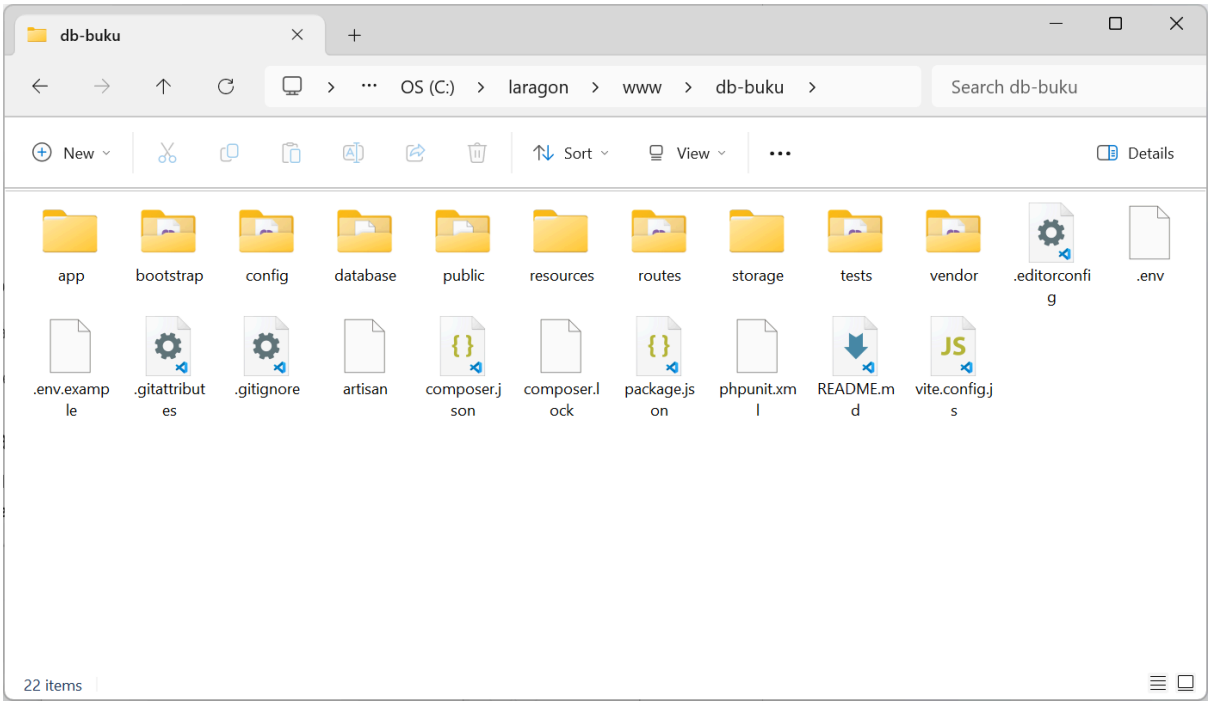
Seterusnya anda akan ditanya nama projek Laravel anda.



Masukkan nama projek dan klik butang OK. Sebagai contoh, masukkan nama projek **db-buku**.

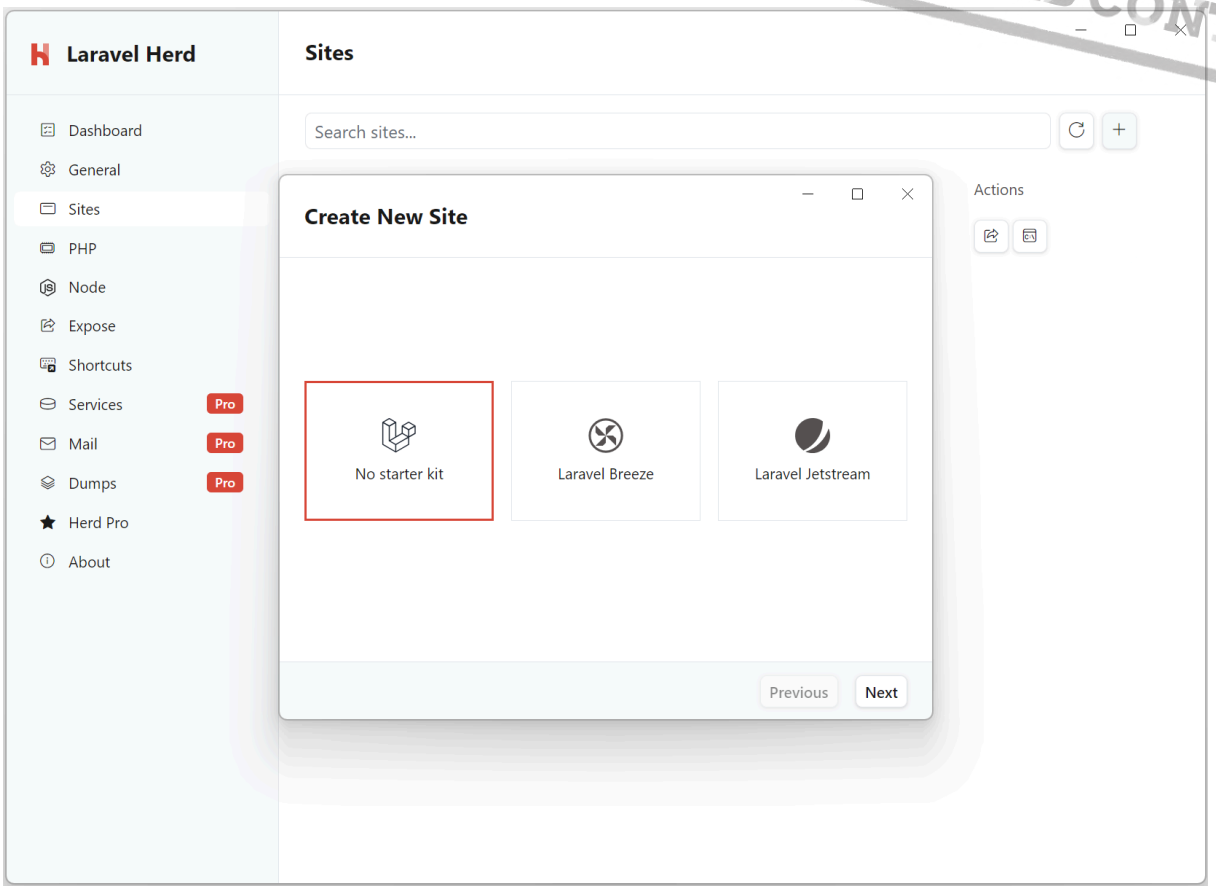
Folder untuk aplikasi boleh diakses dari C:\laragon\www\db-buku.

Aplikasi projek boleh diakses menggunakan pelayar Internet di https://db-buku.test.

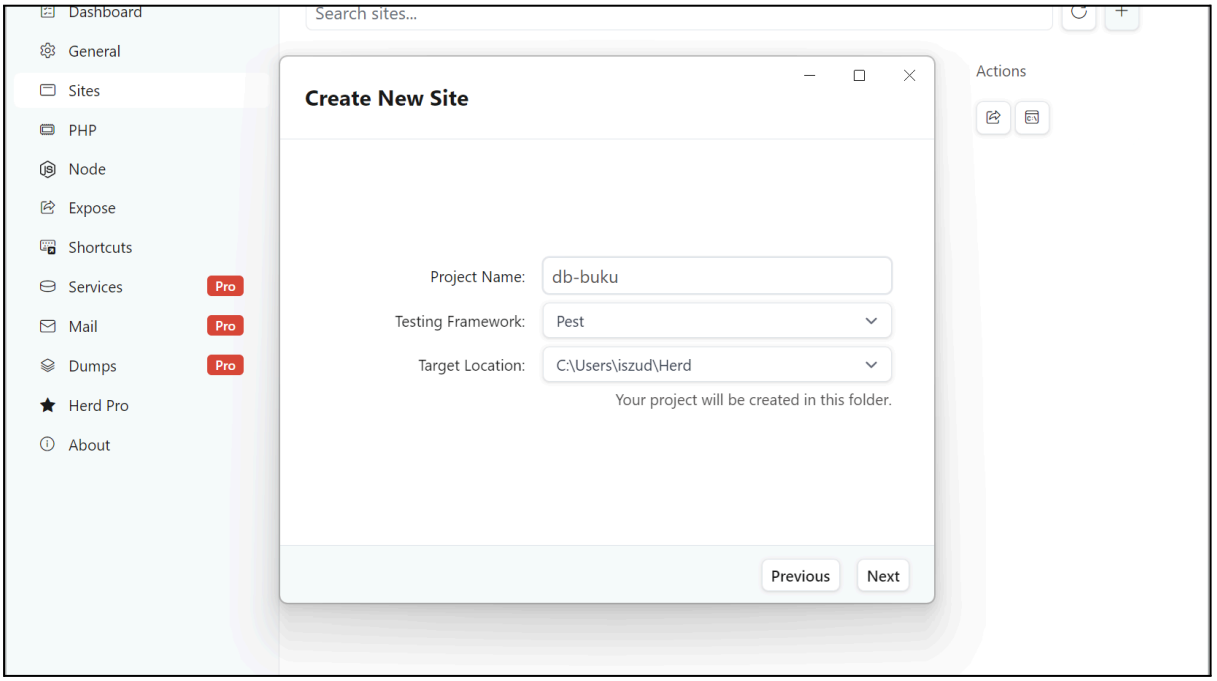


Aplikasi Laravel di Laravel Herd (MacOS)

Dengan Laravel Herd, pergi ke halaman Sites mengguna menu di kiri aplikasi. Pilih ***No starter kit*** dan klik butang ***Next.***

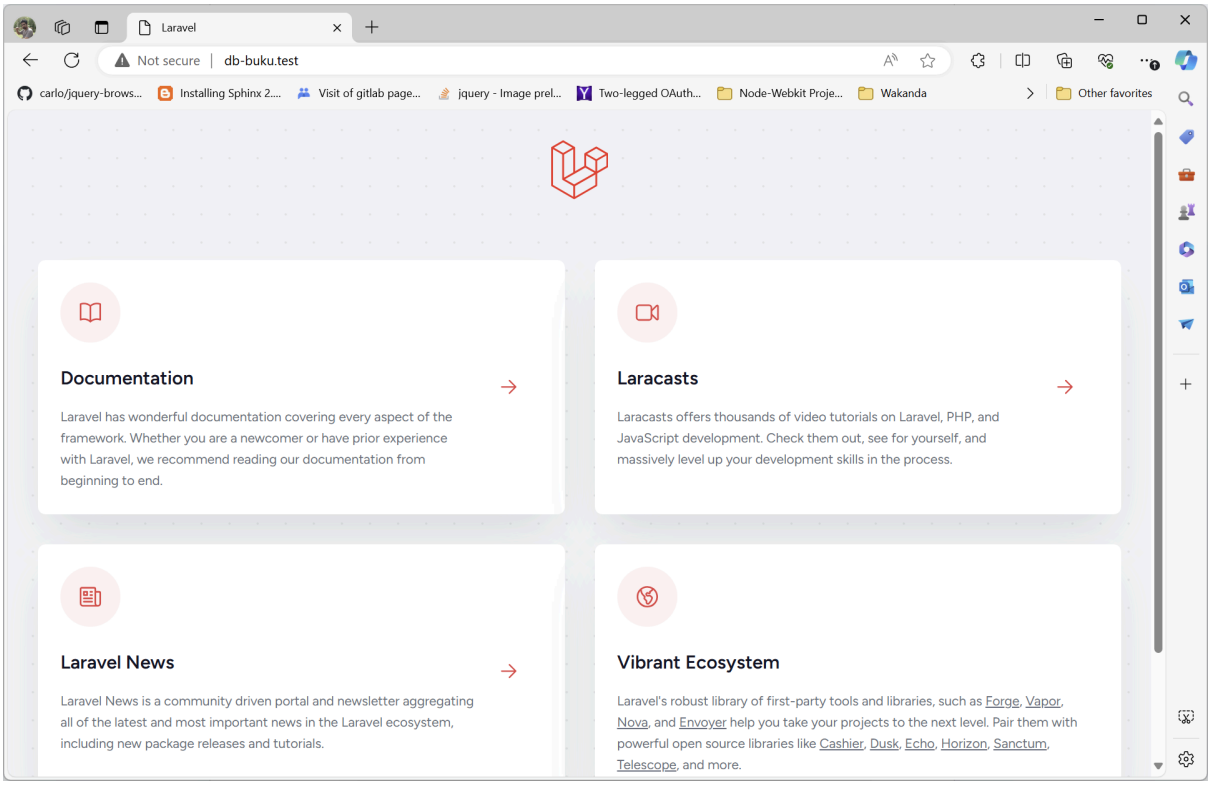


Masukkan nama projek dan klik butang ***Next***.



Folder untuk aplikasi boleh diakses dari C:\Users\<nama pengguna>\Herd\db-buku.

Aplikasi projek boleh diakses menggunakan pelayar Internet di https://db-buku.test.

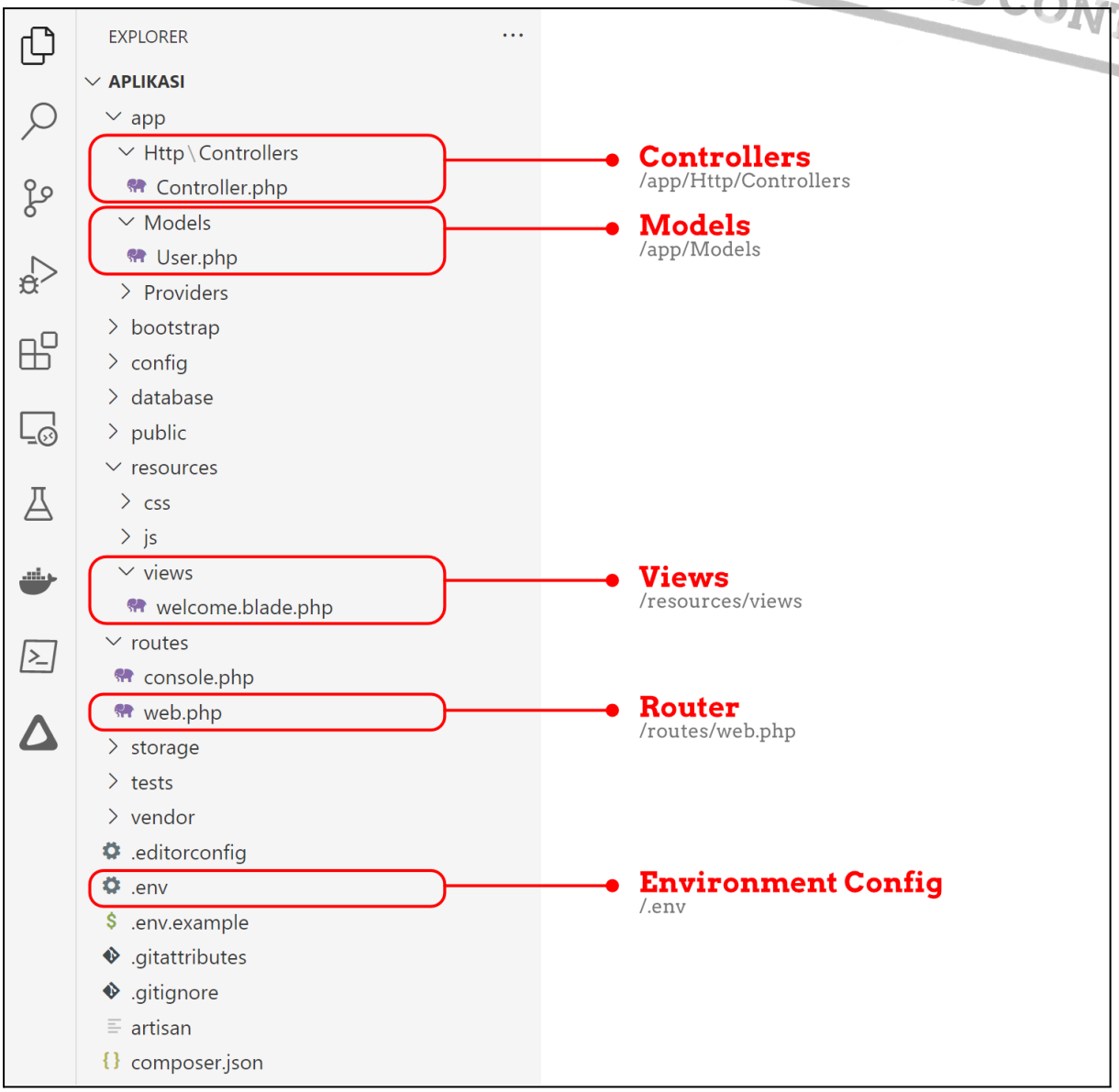


Struktur Fail & Folder

Buka folder aplikasi anda dalam Visual Studio Code. Sekarang kita sudah boleh mula mengubahsuai fail dalam projek Laravel untuk membina aplikasi yang diinginkan.

Aplikasi Laravel yang dipasang menyediakan fail dan folder seperti berikut.

Dalam mempelajari Laravel sebagai sebuah kerangka MVC, perhatikan lokasi di mana *Model*, *View*, *Controller* and juga *Router* disimpan.



Konfigurasi

Laravel perlu ditetapkan beberapa tetapan sebelum kita boleh bermula. Ada dua tempat di mana Konfigurasi boleh dilaksanakan.

1. Fail .env di folder /aplikasi atau paling tinggi dalam projek
2. Fail-fail dalam folder /config

Tetapan di fail .env akan sentiasa diutamakan berbanding tetapan di dalam folder /config. Sekiranya tiada fail .env, gunakan fail .env.example dan tukarkan namanya kepada .env.

Berikut adalah beberapa tetapan utama :

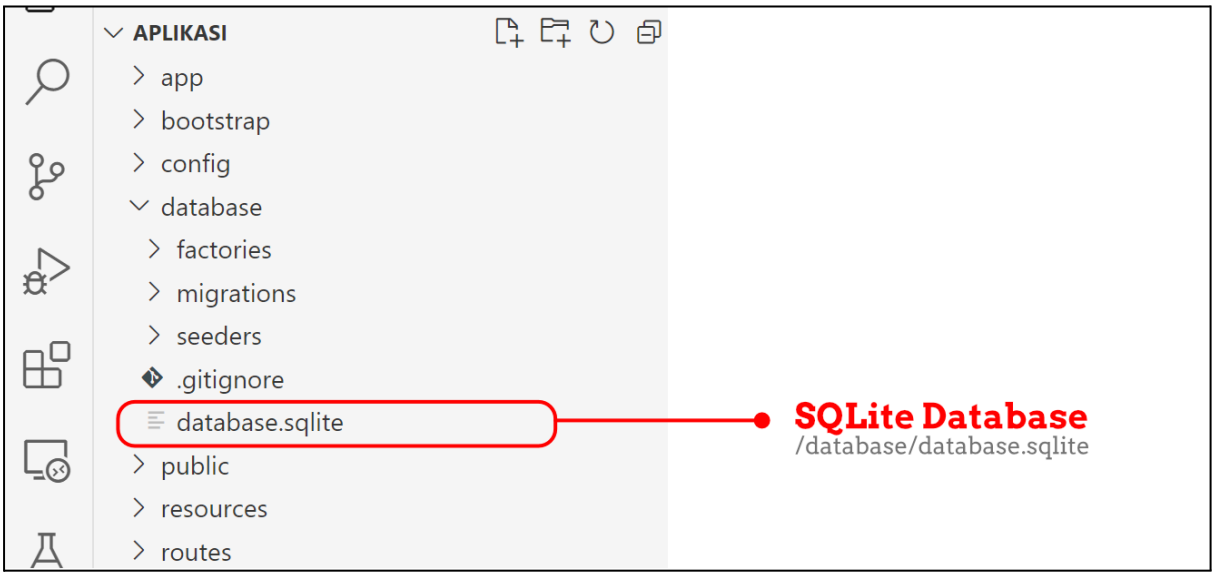
Fail .env

```
APP_NAME=DBBuku
APP_ENV=local
APP_KEY=base64:agQFo5ZkkeKj2EuTNQDbOTaTscXzTYRMj0jRlQDD9DM=
APP_DEBUG=true
APP_TIMEZONE=UTC
APP_URL=http://db-buku.test

APP_LOCALE=en
APP_FALLBACK_LOCALE=en
APP_FAKER_LOCALE=en_US

APP_MAINTENANCE_DRIVER=file
APP_MAINTENANCE_STORE=database

BCRYPT_ROUNDS=12

LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=sqlite
# DB_CONNECTION=mysql
# DB_HOST=127.0.0.1
# DB_PORT=3306
# DB_DATABASE=laravel
# DB_USERNAME=root
# DB_PASSWORD=

SESSION_DRIVER=database
SESSION_LIFETIME=120
SESSION_ENCRYPT=false
SESSION_PATH=/
SESSION_DOMAIN=null

BROADCAST_CONNECTION=log
FILESYSTEM_DISK=local
QUEUE_CONNECTION=database

CACHE_STORE=database
CACHE_PREFIX=

MEMCACHED_HOST=127.0.0.1

REDIS_CLIENT=phpredis
REDIS_HOST=127.0.0.1
```

```
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=log
MAIL_HOST=127.0.0.1
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

VITE_APP_NAME="${APP_NAME}"
```

Pengkalan Data

Secara asalnya, Laravel 11 akan menggunakan database sqlite. Dengan SQLite, file database disimpan di /database/database.sqlite



Tetapan ini boleh diubah sekiranya anda mahu menggunakan MySQL sebagai database. Ubah file .env dengan tetapan seperti berikut :

```
# DB_CONNECTION=sqlite
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=aplikasi
DB_USERNAME=root
DB_PASSWORD=
```

Namun untuk pembangunan, kita akan mengekalkan penggunaan pengkalan data SQLite.

Dengan aplikasi awal Laravel ini, kita sudah boleh mula membina aplikasi kita sendiri.

Di bab seterusnya, kita akan mula menggunakan Laravel dan mula mengenali ciri-cirinya.

Mengenali Laravel

Apa itu MVC?

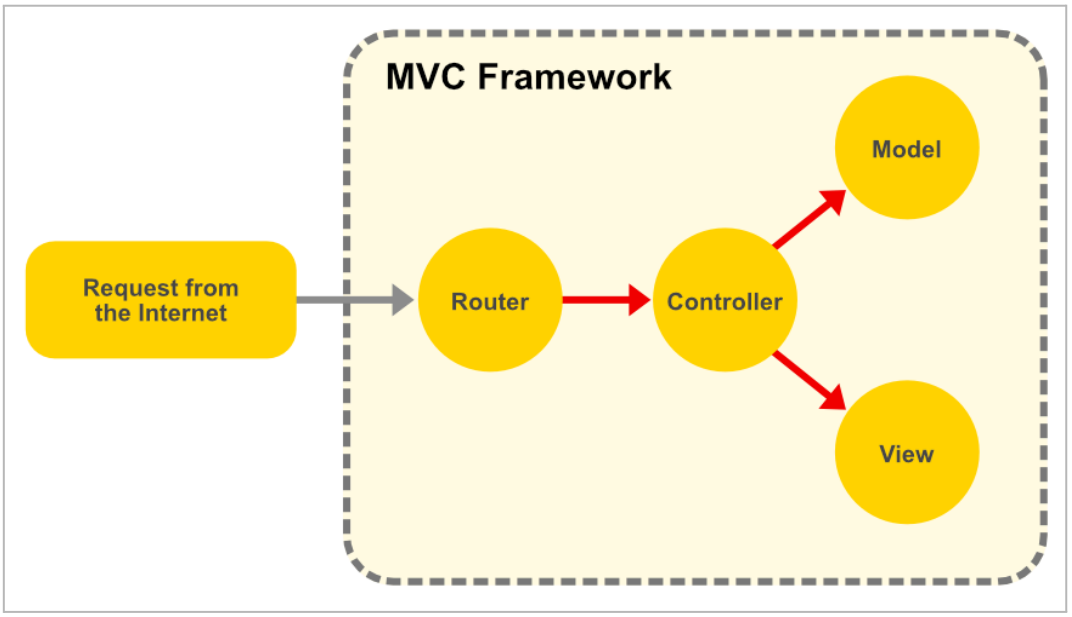
MVC merujuk kepada akronim Model-View-Controller.

- ***Model*** - kod sumber yang menyimpan fungsi-fungsi berkait dengan data dan pengkalan data. Sebuah model merujuk kepada sebuah *table* di dalam sebuah pengkalan data.
- ***View*** - kod sumber yang mengandungi HTML dan kod pengaturcaraan untuk menghasilkan paparan yang diperlukan kepada pengguna. Ia juga boleh menerima data untuk disusun semula diselangselikan dengan kod HTML.
- ***Controller*** - kod sumber yang menjadi pusat pemprosesan permintaan (request) daripada pengguna. *Controller* akan memanggil *Model*, *View* dan lain-lain *library* yang diperlukan.

Dengan pendekatan MVC dan pemisahan fungsi ini, ia menjadi asas kepada bagaimana kod sumber dalam Laravel disusun.

Tetapi ada satu lagi konsep yang perlu diterangkan bersama dengan MVC iaitu Router.

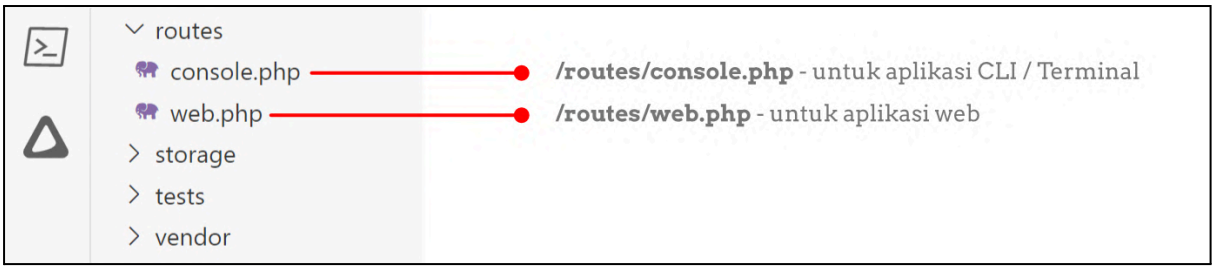
- ***Router*** - kod sumber yang menyimpan tetapan alamat URL dalam aplikasi web yang dibina. Sebuah aplikasi web pastinya mempunyai alamat seperti "/login" atau "/register" dan lain-lain. Setiap satu ini perlu dinyatakan di dalam *Router* bersama lain-lain maklumat berkaitan seperti *Controller* yang akan digunakan dan sebagainya.



Router

Dokumentasi penuh di https://laravel.com/docs/routing.

Router menentukan URL aplikasi dan bagaimana ianya diproses. Fail untuk *Router* boleh didapati di folder /routes.



Untuk pembangunan web, kita hanya gunakan **web.php**.

🗒️**Nota :** Terdapat juga 2 lagi jenis *router* iaitu channel.php dan api.php. Secara asal, kedua-dua ini tidak disertakan bersama seperti versi Laravel yang terdahulu.

Berikut adalah beberapa ciri-ciri dan fungsi *Router* :

1. Redirect
2. View
3. Method / Verb
4. Anonymous function
5. Parameter
6. Controller
7. Model Binding
8. Nama Router
9. Grouping (pengasingan)

Kita akan mempelajari beberapa jenis *Router* ini di sepanjang pelajaran dalam buku ini.

Apabila membuka halaman untuk projek, http://db-buku.test/ ia menggunakan tetapan "/". Buka fail **routes/web.php** untuk menyemak tetapan ini.

Fail routes/web.php

```
<?php

use Illuminate\Support\Facades\Route;
```

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

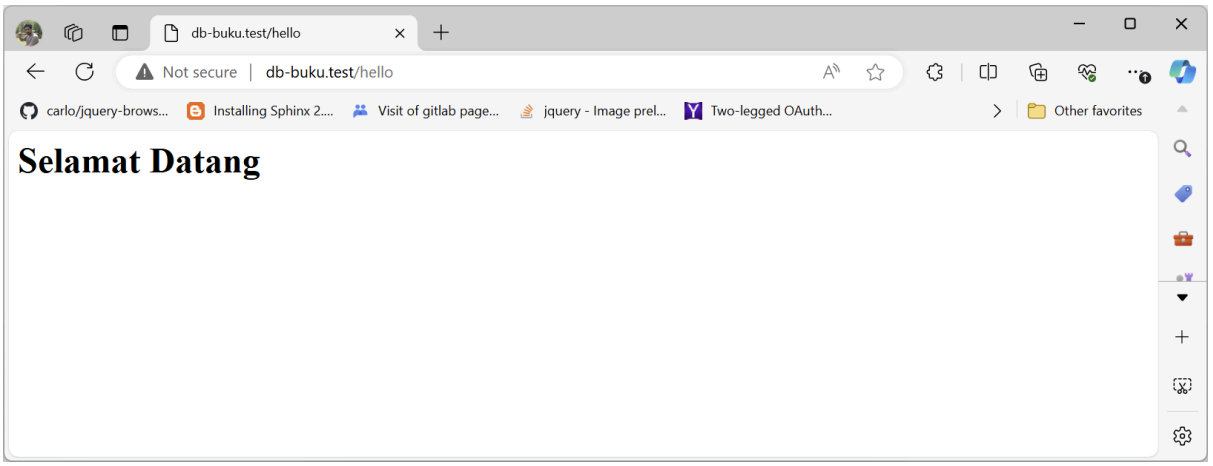
Di sini, halaman utama (/) menggunakan *router* jenis *Method* / *Verb* dengan fungsi **get()** dengan *anonymous function* dan memulang kandungan dari dalam *View* bernama **welcome.blade.php**.

Tambahkan kod berikut di dalam fail **routes/web.php**.

Fail routes/web.php

```
...

Route::get('/hello', function () {
    return "<h1>Selamat Datang</h1>";
});
```

Buka halaman http://db-buku.test/hello.



Controller

Dokumentasi penuh di https://laravel.com/docs/controllers.

Controller menjadi tempat di mana logik dan pemprosesan berlaku.

Selain daripada *anonymous function* yang ditetapkan dalam *Router*, antara fungsi *Router*, kita boleh menetapkan agar ia menggunakan *Controller*. Dalam *Controller*, kita boleh menulis kod yang lebih panjang dan kompleks mengikut keperluan.

Untuk membina sebuah Controller, cara termudah adalah dengan menggunakan *Artisan*.

🗒️**Nota : *Artisan*** adalah perisian yang digunakan dalam Terminal. Kita perlu menaip arahan untuk *Artisan*. *Artisan* mengandungi pelbagai fungsi untuk menyokong dan memudahkan kerja pembangunan dengan Laravel. Antaranya adalah menjana fail dan kod untuk *Controller*, *Model* dan lain-lain lagi. Kita akan banyak menggunakan *Artisan* lagi kelak.

Buka terminal anda. Pastikan anda berada di *folder* projek yang betul. Gunakan arahan **"cd <nama folder>"** untuk masuk ke sesuatu folder. Gunakan arahan **"cd .."** untuk keluar daripada sesuatu folder.

Taip dan laksanakan arahan berikut :

```
php artisan make:controller PageController
```

Sekarang buka fail PageController.php dari *folder* **app/Http/Controllers**. Kemaskini fail tersebut dengan fungsi index() seperti berikut :

Fail app/Http/Controllers/PageController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PageController extends Controller
{
    public function hello($name) {
        echo "<h1>Hello $name</h1>";
```

```
    }
}
```

Router boleh menetapkan agar Controller yang membuat pemprosesan untuk URL.

```
Route::get('<url>', [<NamaController>::class, '<nama_fungsi>']);
```

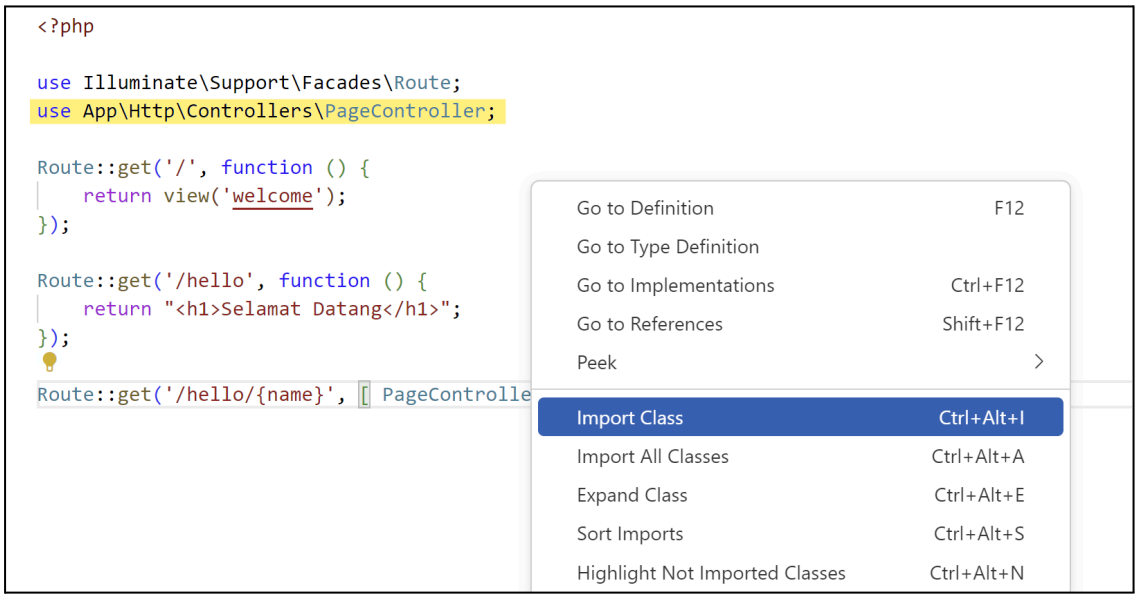
Dan sekarang kita tambahkan kod berikut di dalam fail **routes/web.php**.

Fail routes/web.php

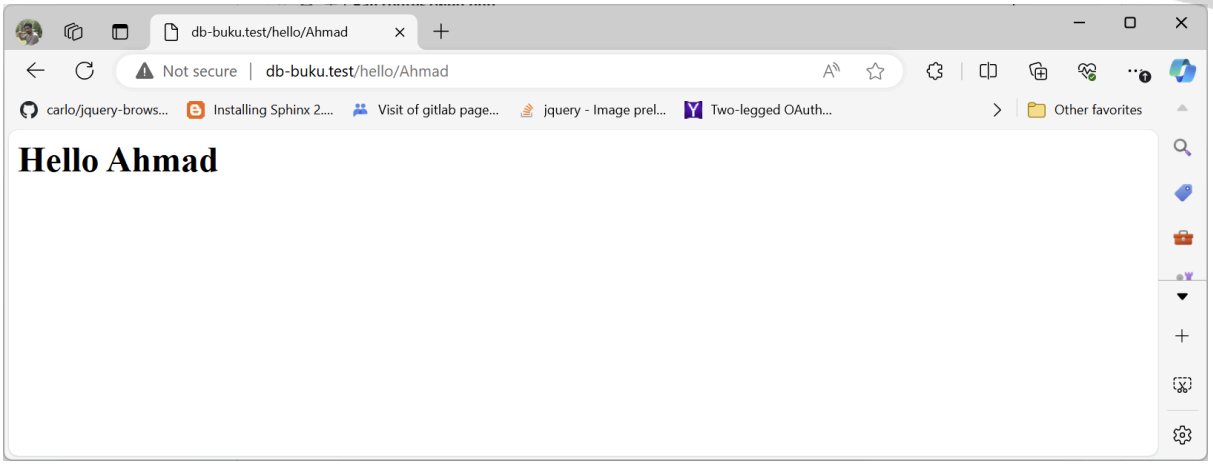
```
...

Route::get('/hello/{name}', [ PageController::class, 'hello']);
```

Apabila kita mula menggunakan kod dari fail berasingan, dalam contoh ini, kita menggunakan kod dari PageController.php di dalam web.php, kita perlu menyatakan keperluan itu terlebih dahulu. Dalam PHP tradisional, kita akan mendapatkan kod dari fail luar dengan arahan require() atau include(). Tetapi dalam pengaturcaraan PHP moden, kita menggunakan *Namespacing*. Mendapatkan fail luar boleh dibuat dengan fungsi **use**. Memandangkan kita sudah memasang beberapa extension untuk memudahkan pengaturcaraan Laravel bersama Visual Studio Code, kita boleh membuat *right-click* pada kod PageController dan memilih **Import Class**. Ini akan menambah kod seperti di bawah di awal fail.

```
use App\Http\Controllers\PageController;
```



Sekarang kita boleh mencuba alamat URL http://db-buku/hello/Ahmad.



Cuba gunakan alamat yang berbeza seperti :

- http://db-buku.com/hello/Raju
- http://db-buku.com/hello/Siti

Daripada contoh di atas, kita dapat melihat bahawa PageController merujuk kepada *class* **PageController** dalam folder **app/Http/Controllers**. Perkataan "hello" pula merujuk kepada nama fungsi di dalam *class* **PageController**.

Router : web.php

```php
Route::get('/hello/{name}', [ PageController::class, 'hello']);
```

Controller : PageController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

2 references | 0 implementations
class PageController extends Controller
{
    1 reference | 0 overrides
    public function hello($name) {
        echo "<h1>Hello $name</h1>";
    }
}
```

💡 Apa yang telah dipelajari?

- ✅ Membina alamat URL untuk aplikasi menggunakan *Router* dan fail web.php.
- ✅ Membina alamat URL dengan *Router* dan *anonymous function*.
- ✅ Cara membina *Controller* dengan Artisan.
- ✅ Menggunakan *Router* dengan fungsi *Controller*.
- ✅ Bagaimana menggunakan data *parameter* dari alamat URL bersama *Controller*.

View & Blade

Dokumentasi penuh di https://laravel.com/docs/blade.

Setakat ini kita telah membina beberapa halaman menggunakan *Router* dan *Controller*. Seterusnya kita akan mula mempelajari *View*.

Dalam Laravel, *Blade* adalah *templating library* yang telah dibangunkan untuk kegunaan bersama *View*. Fungsi View adalah untuk menyusunatur rekaletak halaman web bersama data, sekiranya perlu. Maka View akan mengandungi banyak code HTML dan CSS. Tetapi bersama dengan *Blade*, ia juga akan mengandungi beberapa kod PHP untuk memudahkan integrasi data dan rekaletak.

⚒️ **Tugasan : Mengubah Halaman Utama**

Halaman utama sekarang memaparkan maklumat am tentang Laravel. Kita akan membina sebuah *View* baru untuk menukar rupa bentuk halaman utama.

Kita mulakan dengan membina sebuah *View*. Fail untuk *View* terletak di folder **resources/views**. Di sini kita akan membina sebuah fail View baru dengan nama **home.blade.php**.

🗒️ **Nota :** *View* di dalam Laravel menggunakan *templating library* bernama Blade. Fail untuk View dalam Laravel mesti berakhir dengan **.blade.php**.

Bina fail **home.blade.php** tersebut dengan kandungan HTML seperti demikian.

Fail resources/views/home.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DB Buku</title>
    <style>
        body {
            background-color: #f0f0f0;
            height: 100vh; overflow: hidden;
            display: flex;
            justify-content: center;
            align-items: center;
        }

        #main {
            background-color: aliceblue;
            box-shadow: 0px 0px 20px #00000033;
            border: 2px solid #fff;
            padding: 100px 200px;
            border-radius: 20px;
            display: flex;
            justify-content: center;
            align-items: center;
```

```
            flex-direction: column;
        }

        h1, h2 {
            font-family: Arial, Helvetica, sans-serif;
            margin: 0px; padding: 0px;
        }
    </style>
</head>
<body>
    <div id="main">
        <h1>Selamat Datang</h1>
        <h2>Sistem DB Buku</h2>
    </div>
</body>
</html>
```

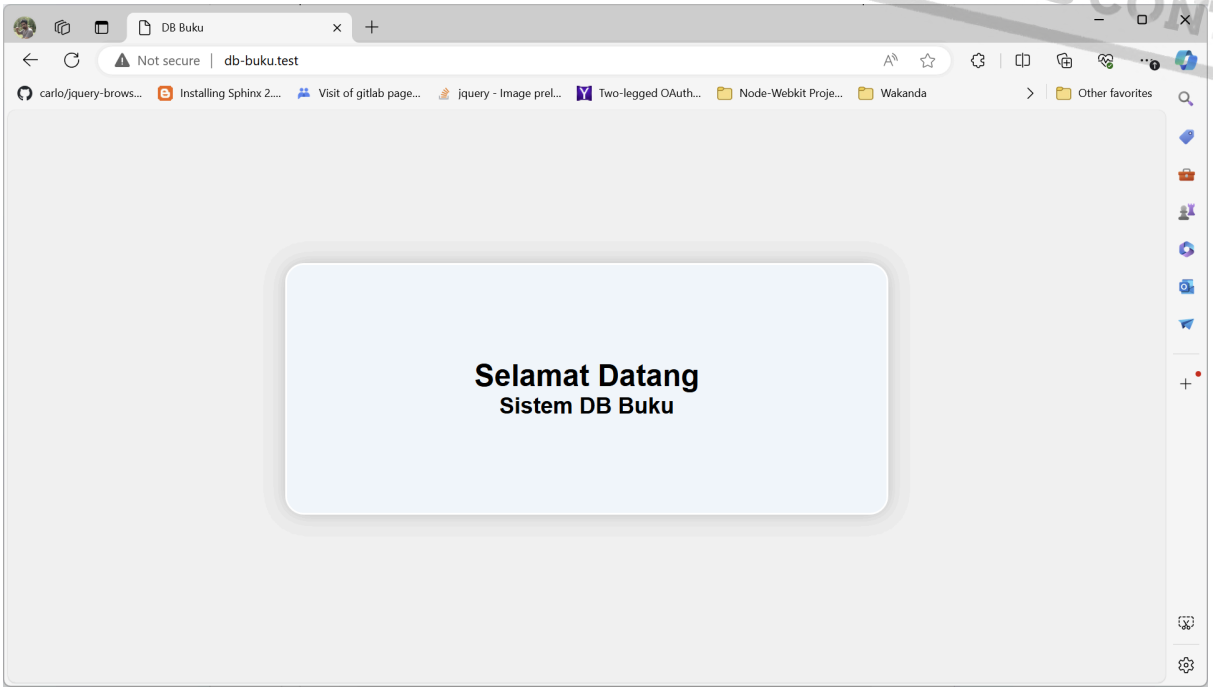
Sekarang kita boleh mengubahsuai fail web.php. Perhatikan bahawa **'welcome'** akan ditukar kepada **'home'** merujuk kepada fail *View* baru **home.blade.php** tadi.

Fail routes/web.php

```
...

Route::get('/', function () {
    return view('home');  // sebelumnya view('welcome')
                          // 'home' merujuk kepada fail
                          // resources/views/home.blade.php
});
```

Cara memanggil fail *View* tidak memerlukan nama extension .blade.php.

Semak laman web utama http://db-buku.test.



💡 Apa yang telah dipelajari?

✅ Bagaimana membina fail view dan di mana ia disimpan.
✅ Bagaimana memanggil fail view untuk dipaparkan.

Kita akan meneruskan pembelajaran View dengan satu lagi tugasan.

⚒️ Tugasan : Membina Laman Web Ringkas

Kita akan membina sebuah laman web yang mempunyai halaman berikut :

- Utama
- Mengenai Kami
- Hubungi

Terdapat menu di bahagian atas untuk beralih dari satu laman ke laman yang lain. Setiap halaman mempunyai rekaletak yang sama dengan kandungan yang berbeza.

Kita mulakan dengan menetapkan *Router*. Buka fail **web.php** dan tambahkan kod sedemikian.

Fail routes/web.php

```
...

Route::get('/page/{page}', [PageController::class, 'index']);
```

Parameter {page} akan bertukar menjadi *variable* ***$page*** di dalam PageController kelak. Dengan nilai *variable* ini, kita akan memanggil fail *View* yang bersesuaian. Apa yang kita cuba hasilkan adalah di mana kita akan ada alamat URL seperti berikut yang dipadankan kepada fail *View*.

Perhatikan bahawa kita akan membina satu folder khusus untuk himpunan View baru ini.

Alamat URL	Fail View
/page/**utama**	resources/views/pages/**utama**.blade.php
/page/**mengenai**	resources/views/pages/**mengenai**.blade.php
/page/**hubungi**	resources/views/pages/**hubungi**.blade.php

Kita juga akan membina satu fail View untuk reka letak keseluruhan laman web. Ini akan kita letakkan dalam *folder* **layout** dengan nama **page.blade.php**. Kita mulakan dengan membina fail **resources/views/layout/page.blade.php**.

Fail resources/views/layout/page.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ $title ?? 'DB Buku' }}</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+
ALEwIH" crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N
6jIeHz" crossorigin="anonymous"></script>
</head>
<body>
    <div class="container mt-5">
        <div id="header" class="row">
            <div class="col">
```

```
                <ul class="nav nav-pills border-bottom pb-3 border-1
border-bottom">
                    <li class="nav-item">
                        <a class="nav-link" href="/page/utama">Utama</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/page/mengenai">Mengenai
Kami</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/page/hubungi">Hubungi</a>
                    </li>
                </ul>
            </div>
        </div>

        <div class="row mt-5">
            <div class="col" id="content">
                @yield('content')
            </div>
        </div>
    </div>
</body>
</html>
```

🗒️ **Nota : Memaparkan nilai dan *Blade Directive***

Terdapat dua operasi penting di dalam View. Kita perlu memaparkan nilai yang dihantar kepadanya, dan membuat beberapa operasi.

Blade Directive	Ini adalah fungsi khas yang digunakan di dalam *Blade View.* Ia bermula dengan aksara **@** seperti `@yield()`, `@section()`, dan `@extends`.
Memaparkan Nilai	Memaparkan nilai akan menggunakan aksara **`{{ ... }}`**

Sekarang kita boleh mula membina View untuk setiap kandungan. Mulakan dengan membina *folder* **pages** di dalam *folder* **resources/views**.

Fail resources/views/pages/utama.blade.php

```
@extends('layout.page')

@section('content')
    <h1>Utama</h1>

    <p>In sit autem totam id recusandae reiciendis non
    sapiente minima sunt, voluptas, nobis soluta
    nihil, consequatur voluptatibus!</p>
@endsection
```

Fail resources/views/pages/hubungi.blade.php

```
@extends('layout.page')

@section('content')
    <h1>Hubungi</h1>

    <p>Lorem ipsum dolor sit amet consectetur
    adipisicing elit. Nostrum maiores
    labore vero, ea sunt facilis natus quae.</p>
@endsection
```

Fail resources/views/pages/mengenai.blade.php

```
@extends('layout.page')

@section('content')
    <h1>Mengenai Kami</h1>

    <p>Placeat inventore, maiores modi similique
    aperiam totam rem nulla culpa, est
    laboriosam molestiae rerum tenetur dicta!</p>
@endsection
```

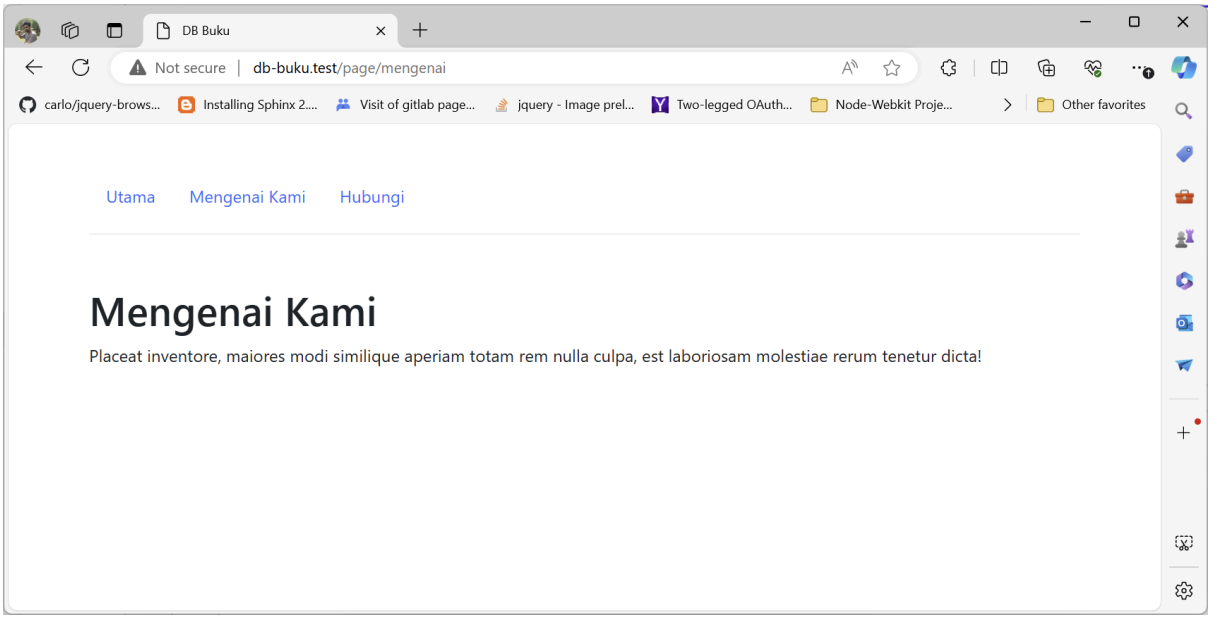
Sekarang kita boleh mengemaskini PageController dengan fungsi index().

Fail app/Http/Controllers/PageController.php

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

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

class PageController extends Controller
{
    public function hello($name) {
```

```
        echo "<h1>Hello $name</h1>";
    }

    public function index(Request $request, $page) {
        $page_view = "pages.$page";
        if (View::exists($page_view)) {
            return view($page_view);
        }
        abort(404);
    }
}
```

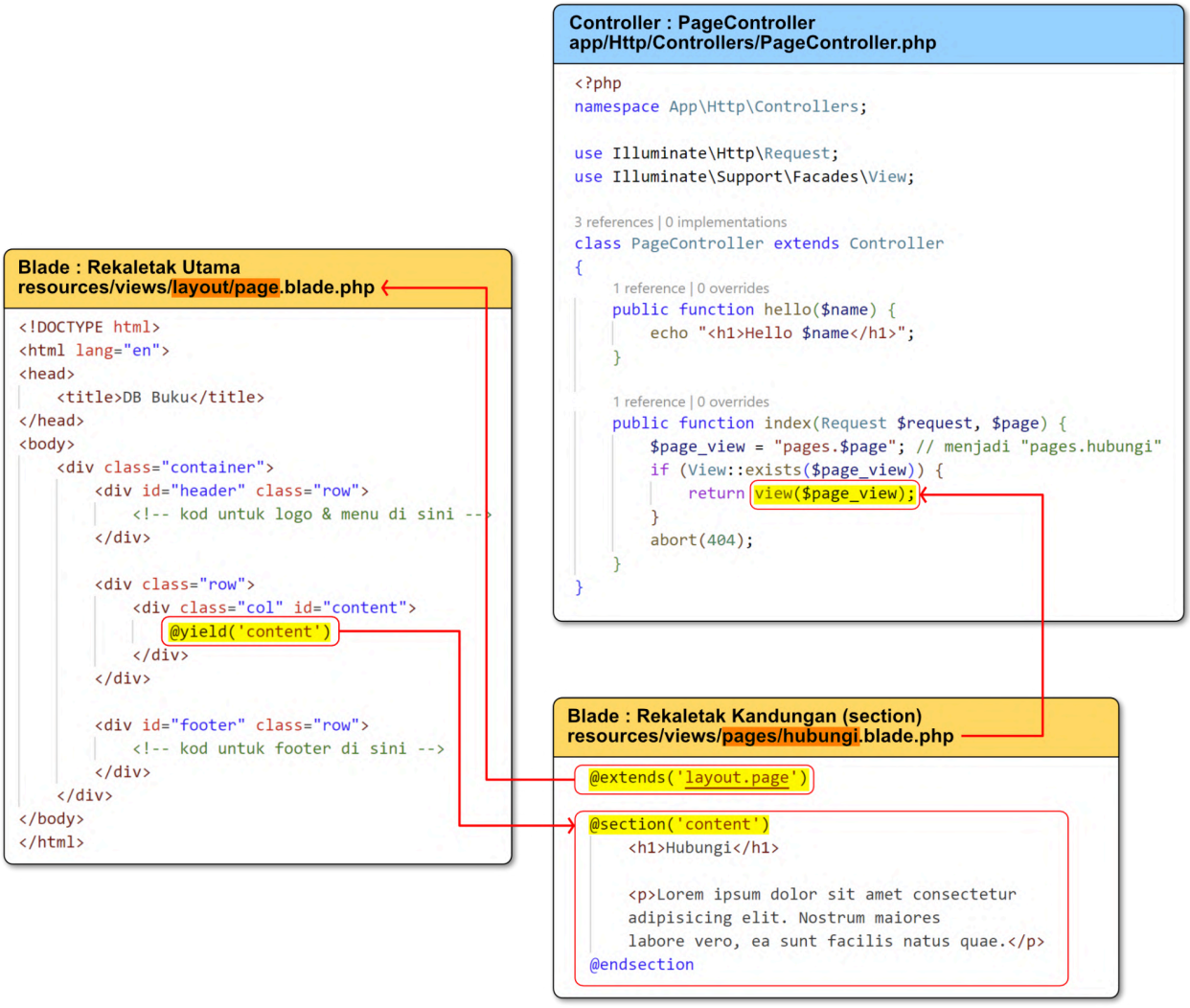
Sekarang kita boleh menguji laman web baru ini dengan alamat http://db-buku.test/page/utama.



Berikut adalah gambaran bagaimana fail-fail view berhubung-kait menggunakan **@layout()**, **@section()** dan **@extends()**.

🗒️**Nota :** Apabila fail view blade berada di dalam folder yang lebih dalam dari resources/views, memanggilnya dari Controller perlu menggunakan aksara titik (.) untuk menandakannya. Perhatikan contoh di bawah :

Fail *view blade*	**Dipanggil dalam *Controller***
`resources/views/pages/home.blade.php`	`return view('pages.home');`
`resources/views/layout/main.blade.php`	`return view('layout.main');`



💡 Apa yang telah dipelajari?

- ✅ Fungsi *Layout* dan *Section* dalam *Blade View* menggunakan **@yield()**, **@section()** dan **@extends()**.
- ✅ Memeriksa kewujudan fail *View* dengan **View::exists()**
- ✅ Memanggil *View* dari dalam *Controller*.

Kita sambung pembelajaran View dengan sedikit lagi tugasan.

⚒️ Tugasan : Mengemaskini Menu

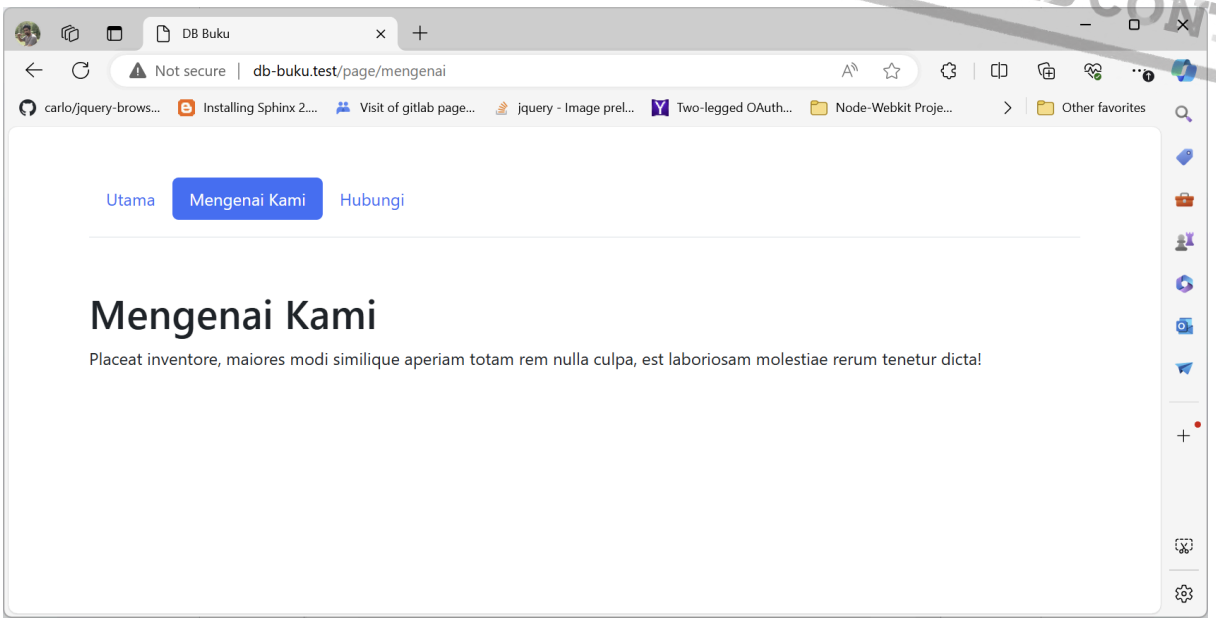
Laman web yang baru dibina tidak menunjukkan di menu di halaman mana kita berada. Menggunakan *class* **active** yang didatangkan bersama Bootstrap CSS, kita akan menunjukkan pada menu di halaman mana kita berada.

Kemaskini fail page.blade.php dengan kod seperti di bawah :

Fail resources/views/layout/page.blade.php

```
. . . .
                <ul class="nav nav-pills border-bottom pb-3 border-1
border-bottom">
                    <li class="nav-item">
                        <a class="nav-link {{ (request()->is('page/utama'))
? 'active' : '' }}" href="/page/utama">Utama</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link {{
(request()->is('page/mengenai')) ? 'active' : '' }}"
href="/page/mengenai">Mengenai Kami</a>
                    </li>
                    <li class="nav-item}">
                        <a class="nav-link {{
(request()->is('page/hubungi')) ? 'active' : '' }}"
href="/page/hubungi">Hubungi</a>
                    </li>
                </ul>
. . . .
```

Dalam kod di atas, kita menggunakan *Helper* **request()** untuk mengenalpasti alamat URL semasa. Dengan berikut kita boleh menambah *class* **active** sekiranya benar. Dengan *class* **active** ini, item menu tersebut akan mempunyai latar berwarna biru.



💡 **Apa yang telah dipelajari?**

✅ Menggunakan fungsi ***request()->is()*** di dalam fail *blade* untuk mengenalpasti URL semasa.
✅ Cara menambah class CSS secara dinamik dalam di dalam fail *blade*.

Sehingga kini, dengan apa yang telah dibina, kita boleh membina sebuah fail *view blade* baru di dalam folder **resources/views/pages** dan ianya boleh diakses terus melalui *URL* **http://db-buku.test/pages/<nama-view>**.

Pengkalan Data & Model

Sehingga kini, kita masih belum menggunakan pengkalan data. Untuk berinteraksi dengan pengkalan data, kita akan menggunakan Model. Model adalah sebahagian daripada konsep kerangka MVC.

Bahagian ini merupakan satu demonstrasi dan pengenalan kepada Model dan operasi pengkalan data. Kita akan membincangkan tentang model di dalam bab yang lebih khusus kemudian.

Konfigurasi Pengkalan Data

Sebelum melaksanakan operasi pengkalan data, kita kena memastikan sistem boleh membuat sambungan ke pengkalan data. Untuk bab ini, kita akan menggunakan pengkalan data SQLite seperti yang diterangkan dalam topik ***Pengkalan Data*** dalam bab ***Membina Aplikasi Laravel.***

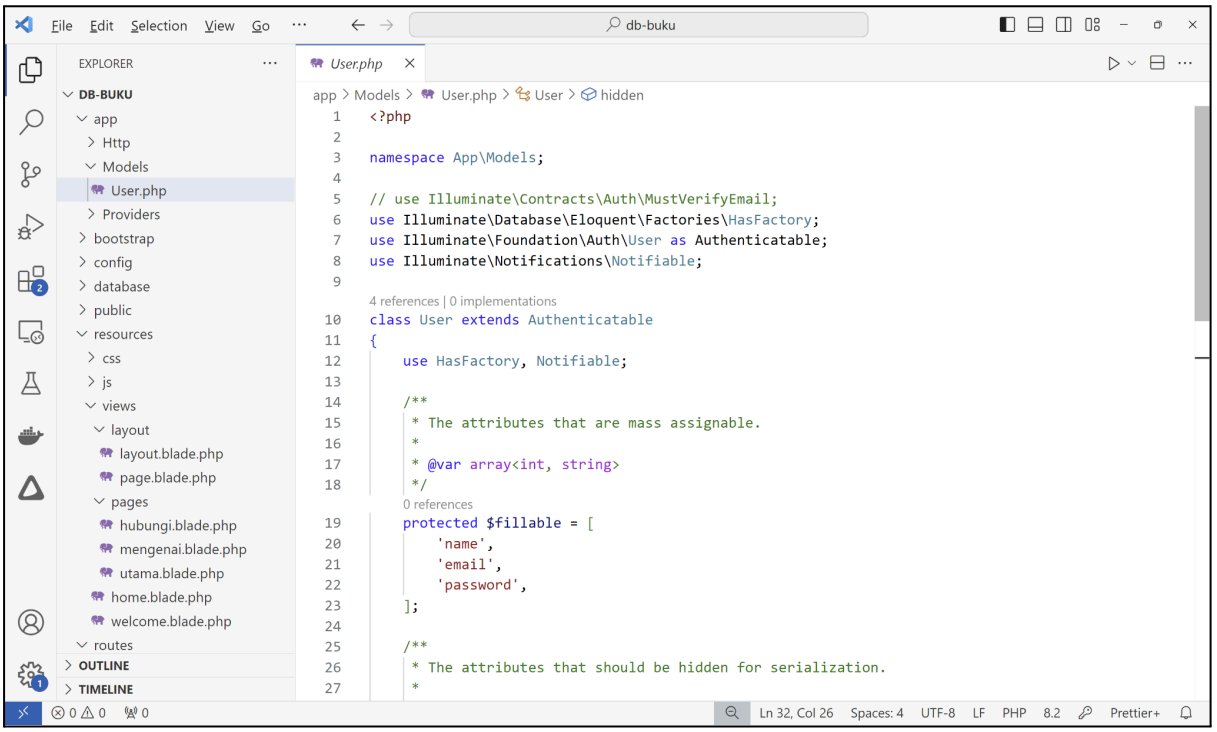
Pastikan fail konfigurasi .env anda mempunyai tetapan seperti berikut :

Fail .env

```
...

DB_CONNECTION=sqlite
# DB_CONNECTION=mysql
# DB_HOST=127.0.0.1
# DB_PORT=3306
# DB_DATABASE=laravel
# DB_USERNAME=root
# DB_PASSWORD=

...
```

User Model

Laravel didatangkan dengan satu model iaitu *User* Model. *User* Model ini terdapat di **app/Models/User.php**.



Sesebuah model itu merujuk kepada sebuah *table* di dalam pengkalan data. Untuk ***User*** model ini, ia merujuk kepada *table* **users**. Tapi mungkin *table* ini belum wujud lagi. Maka kita akan melaksanakan satu arahan *Artisan* untuk membina *table* ini.

Buka ***Terminal*** dan pastikan anda di *folder* projek yang betul. Laksanakan arahan berikut :

```
cd db-buku <Enter>

php artisan migrate <Enter>
```

Nota : Arahan Asas Linux untuk Kegunaan dalam Terminal

- `cd <nama-folder>` : Masuk ke dalam *folder*
- `dir` : Menyenaraikan kandungan *folder* semasa
- `ls -al` : Menyenaraikan kandungan *folder* semasa
- `pwd` : Mendapatkan lokasi semasa
- `cd ..` : Keluar dari *folder*

Arahan di atas akan membantu membina beberapa *table* dalam pengkalan data termasuk *table* **users**. Sekarang kita boleh mula memasukkan berapa data contoh dalam *table* user.

Kita akan menggunakan satu lagi ciri dalam Laravel yang dipanggil **Database Seeder**. Kita akan membincangkan *Database Seeder* dengan lebih terperinci dalam bab yang lain. Untuk bab ini, kita akan menggunakannya untuk menjana beberapa data contoh untuk *table* **users**.

Buka fail **database/seeders/DatabaseSeeder.php**.

Kemaskini fail supaya baris 18 hingga 21 dikomen, dan buang komen pada baris 16.

Fail database/seeders/DatabaseSeeder.php

```
. . .
14      public function run(): void
15      {
16          User::factory(10)->create();
17
18          // User::factory()->create([
19          //     'name' => 'Test User',
20          //     'email' => 'test@example.com',
21          // ]);
22      }
. . .
```

Apabila kita melaksanakan arahan Artisan Seeder nanti, ia akan mengambil arahan daripada fail DatabaseSeeder.php ini dan menjana 10 rekod contoh untuk *table* **users**.

Kembali ke Terminal dan laksanakan arahan berikut :

```
php artisan db:seed
```

Sekarang kita boleh mula menggunakan User model untuk memaparkan rekod dari dalam **users** *table*.

Membina Paparan Senarai Pengguna

⚒️ Tugasan : Membina Paparan Senarai Pengguna

Dalam tugasan ini, kita akan melaksanakan perkara-perkara berikut :

- Membina sebuah *Controller* baru **UserController.php** menggunakan **Artisan**
- Mengemaskini *Controller* untuk mendapatkan data menggunakan **User** Model.
- Membina sebuah *view blade* baru di **resources/users/index.blade.php**
- Menambah tetapan router baru untuk /users dalam fail web.php

Satu lagi fungsi Artisan adalah untuk membantu menjana kod. Kita boleh menjana kod untuk sebuah *Controller* dengan mudah menggunakan **Artisan**. Laksanakan arahan berikut di Terminal.

```
php artisan make:controller UserController
```

Sebuah fail telah dijana. Sekarang buka fail dari app/Http/Controllers/UserController.php. Kemaskini fail tersebut dengan menambah fungsi index() seperti kod di bawah :

Fail app/Http/Controllers/UserController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index() {
        $users = User::all(); // Menggunakan User model untuk mendapatkan
                              // kesemua rekod dari table users
```

```
        return view('users', compact('users')); // data $users dipanjangkan
                                                // ke view users.blade.php
    }
}
```

Perhatikan kod yang bertanda kuning di atas.

Dalam kod di atas, kita mula menggunakan *User* model. Seperti mana kita pernah gunakan Namespacing dengan use di dalam web.php *(rujuk bahagian Controller yang lepas)*, kita perlu lakukan yang sama mendapatkan kod dari **User.php** di folder **app/Models**.

Sekarang kita boleh membina fail *view blade* untuk menyenaraikan rekod dari *users* table.

Bina fail **users.blade.php** di dalam folder **resources/views**.

Fail resources/views/users.blade.php

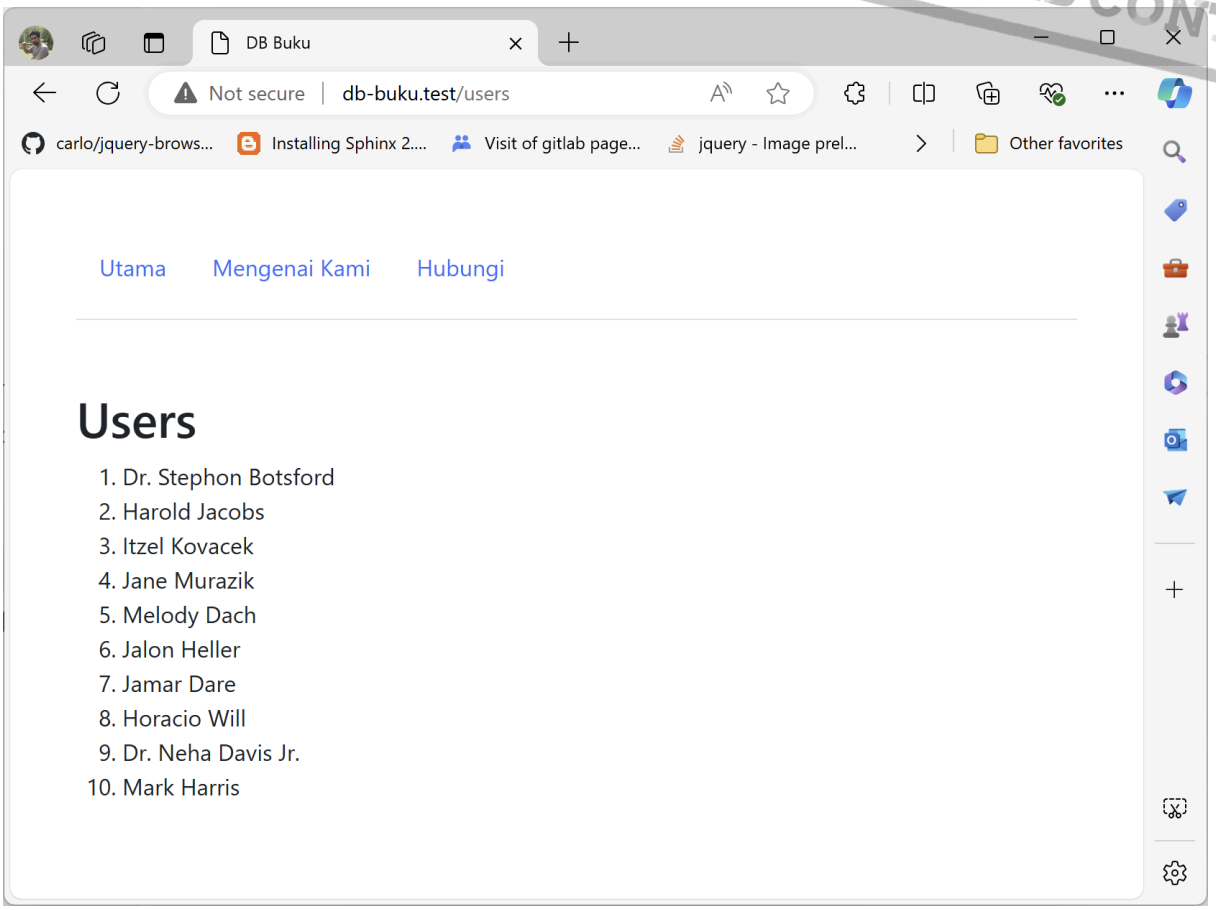
```
@extends('layout.page')

@section('content')
    <h1>Users</h1>

    <ol>
    @foreach($users as $user)
        <li>{{ $user->name }}</li>
    @endforeach
    </ol>
@endsection
```

Akhir sekali, kita perlu mengemaskini router web.php dengan URL baru.

Fail routes/web.php

```
<?php
. . .
use App\Http\Controllers\UserController;
. . .
Route::get('/users', [UserController::class, 'index']);
```

Sekarang kita boleh menguji halaman baru yang telah dibina. Kod di atas juga menggunakan kod dari fail **UserController.php** yang berada di folder **app/Http/Controllers**. Pastikan anda menggunakan fungsi **use** untuk mendapatkan kod dari fail tersebut.



🗒️**Nota :** Senarai nama anda dari table users ini mungkin berbeza daripada paparan di atas. Ini kerana *Database Seeder* menggunakan sebuah *library* yang menjana nama secara rawak. Ia berkebarangkalian berbeza daripada paparan di atas.

💡Apa yang telah dipelajari?

- ✅ Bagaimana menjana fail *Controller* dengan Artisan *(php artisan make:controller)*.
- ✅ Pengenalan kepada *Database Migration (php artisan migrate)*.
- ✅ Pengenalan kepada *Database Seeder (php artisan db:seed)*.
- ✅ Cara menggunakan Model untuk mendapatkan kesemua rekod dengan fungsi **all().**
- ✅ Cara menghantar data ke view dengan fungsi **compact()**
- ✅ *Directive Blade* **@foreach** dan **@endforeach** untuk membina gelung dalam **View**.

Penutup

Edisi Bab Contoh 24 Jam Laravel ini memberikan beberapa bab awal sebagai apa yang bakal terbit kelak.

Di dalam beberapa bab yang telah disampaikan ini, saya cuba untuk memberikan permulaan yang pantas kepada Laravel. Harapannya, pembaca dapat merasai sendiri apa itu Laravel dan mula menggunakannya.

Dalam bab-bab mendatang, saya akan perincikan lagi topik-topik berkenaan Namespace, Autoloader, View, Model, Controller, Tinker, Database Migration, Database Seeder, Database Factory dan lain-lain lagi.

Saya mengharapkan sokongan pembaca Edisi Bab Contoh ini untuk menyokong usaha kami membantu mereka yang ingin mempelajari Laravel di Malaysia, untuk mendapatkan satu sumber pembelajaran yang disampaikan dalam bahasa Melayu.

Untuk membantu usaha ini, harap pembaca dapat membuat pra-tempahan kepada edisi bercetak kami yang bakal terbit tidak lama lagi.

Edisi bercetak ini dianggarkan akan mempunyai 350 mukasurat, dan akan dijual pada harga RM89.90.

Namun untuk pra-tempahan, ia bermula dari RM40 sahaja.

Buat pra-tempahan anda hari ini
https://kelasprogramming.com/24jl

Iszuddin Ismail
Penulis