KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
http://www.polinema.ac.id

Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)

JOBSHEET 02
ROUTING, CONTROLLER, DAN VIEW

1. MVC pada Laravel

MVC merupakan singkatan dari Model View Controller. Laravel menggunakan model MVC, oleh karena itu ada tiga bagian inti dari framework yang bekerja bersama: model, view, dan controller. Controller adalah bagian utama di mana sebagian besar pekerjaan dilakukan. Controller terhubung ke Model untuk mendapatkan, membuat, atau memperbarui data dan menampilkan hasilnya pada View, yang berisi struktur HTML aktual dari aplikasi.

a. Model
   Dalam Laravel, kelas Model berisi semua metode dan atribut yang diperlukan untuk berinteraksi dengan skema database yang ditentukan.
b. View
   View mewakili bagaimana informasi ditampilkan, digunakan untuk semua logika antarmuka pengguna perangkat lunak. View mewakili Antarmuka Pengguna (Frontend) dari halaman web.
c. Controller
   Controller berperan sebagai perantara antara Model dan View, memproses semua masukan yang dikirim oleh pengguna dari View. Controller memproses semua logika bisnis, memanipulasi data menggunakan komponen Model, dan berinteraksi dengan View untuk merender output akhir.

2. Routing

Pada Laravel terdapat fitur yang Bernama *route*. Route ini digunakan sebagai penghubung antara user dengan aplikasi. Dengan kata lain, URL yang kita tulis di dalam browser akan melewati route. Dan pada route tersebut akan ditentukan kemana selanjutnya, bisa ke Controller atau ke View.

Routing sendiri adalah proses pengiriman data maupun informasi ke pengguna melalui sebuah permintaan yang dilakukan kepada alamat yang sudah terdaftar, lalu alamat tersebut

akan memproses dari permintaan kita tadi. Setelah proses selesai maka akan mengembalikan sebuah output atau hasil dari proses tersebut.

Untuk membuat route digunakan **Facade Route** diikuti dengan verb yang merupakan **HTTP verb**, umumnya terdiri dari **get, post, put, delete, options, patch**. Selain itu dibutuhkan **path** yang berupa URL setelah nama domain aplikasi yang diakses oleh pengguna. Dan pada bagian akhir terdapat **callback** yang dapat berupa **callback function** atau **controller action** yang menjalankan logika ketika path diakses oleh pengguna.

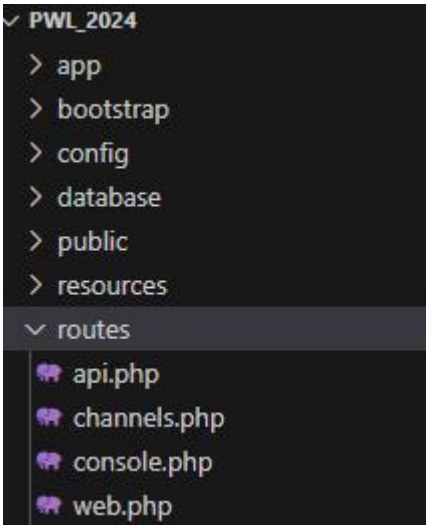
Berikut contoh sederhana penulisan route di Laravel 10.

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

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

Route di atas dapat diterjemahkan ketika pengguna mengakses URL pada **/hello** akan mengeksekusi callback function yang menampilkan pesan 'Hello World'.

Akan tetapi penggunaan callback function jarang sekali dipakai dalam pembuatan aplikasi sesungguhnya, karena untuk logika yang kompleks menjadikan kode susah di-maintenance. Sebagai solusi diperkenalkan konsep Controller. Jika route di atas dikonversi ke controller menjadi sebagai berikut:

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

Route::get('/hello', [WelcomeController::class,'hello']);
```



Di dalam project Laravel, terdapat folder **routes**. Secara umum laravel membagi menjadi empat tempat, yaitu:

a. `routes/web.php` digunakan untuk web standard
b. `routes/api.php` digunakan untuk web service/API
c. `routes/console.php` digunakan untuk command line
d. `routes/channel.php` digunakan untuk broadcast channel melalui websocket

Secara umum aplikasi yang dibuat cukup dengan `routes/web.php` dan `routes/api.php`. Bahkan jika aplikasi tidak perlu menyediakan API hanya menggunakan `routes/web.php` saja.

Pada Laravel kita dapat menggunakan semua http verb untuk dipasang sebagai method router yang ingin digunakan, sudah dijelaskan sebelumnya bahwa semua http verb dapat dilayani dengan Router pada laravel. Endpoint / url router sebaiknya mengikuti best practice berikut ini dimana sebuah resource dapat dilayani dengan fungsi berbeda pada setiap http verb nya.

Resource	POST	GET	PUT	DELETE
/mahasiswa	Membuat record mahasiswa baru	Mengambil Daftar Mahasiswa	Update banyak data mahasiswa	Delete banyak data mahasiswa
/mahasiswa/{id}	Error	Tampilkan Data Satu Mahasiswa	Update data mahasiswa jika ada data dengan id yang dikirim	Delete satu data mahasiswa

Perlu diketahui laravel dapat mendukung satu route yang memiliki lebih dari satu http verb atau memiliki semua http verb. Berikut ini kode program routing untuk tabel di atas

```
Route::get('mahasiswa', function ($id) {
});
Route::post('mahasiswa', function ($id) {
});
Route::put('mahasiswa', function ($id) {
});
Route::delete('mahasiswa', function ($id) {
});
Route::get('mahasiswa/{id}', function ($id) {
});
Route::put('mahasiswa/{id}', function ($id) {
});
Route::delete('mahasiswa/{id}', function ($id) {
});
```

Untuk memeriksa dan memvalidasi apakah route yang dibuat sudah benar dengan menggunakan perintah berikut

```
php artisan route:list
```

Output dari perintah tersebut jika routing yang anda buat benar akan menjadi seperti ini

```
+--------+----------+----------------+------+---------+------------+
| Domain | Method   | URI            | Name | Action  | Middleware |
+--------+----------+----------------+------+---------+------------+
|        | GET|HEAD | /              |      | Closure | web        |
|        | GET|HEAD | api/user       |      | Closure | api        |
|        |          |                |      |         | auth:api   |
|        | GET|HEAD | mahasiswa      |      | Closure | web        |
|        | POST     | mahasiswa      |      | Closure | web        |
|        | PUT      | mahasiswa      |      | Closure | web        |
|        | DELETE   | mahasiswa      |      | Closure | web        |
|        | GET|HEAD | mahasiswa/{id} |      | Closure | web        |
|        | PUT      | mahasiswa/{id} |      | Closure | web        |
|        | DELETE   | mahasiswa/{id} |      | Closure | web        |
+--------+----------+----------------+------+---------+------------+
```

Jika anda membutuhkan route yang dapat memiliki lebih dari satu http method routing nya dapat dibuat dengan cara seperti ini.

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

Route::any('/specialMahasiswa', function ($id) {
});
```

```
+--------+-----------------------------------------+-----------------+------+---------+------------+
| Domain | Method                                  | URI             | Name | Action  | Middleware |
+--------+-----------------------------------------+-----------------+------+---------+------------+
|        | GET|HEAD                                | /               |      | Closure | web        |
|        | GET|HEAD                                | api/user        |      | Closure | api        |
|        |                                         |                 |      |         | auth:api   |
|        | GET|HEAD|POST|PUT|PATCH|DELETE|OPTIONS | specialMahasiswa|      | Closure | web        |
|        | GET|POST|HEAD                           | specialUrl      |      | Closure | web        |
+--------+-----------------------------------------+-----------------+------+---------+------------+
```

- **Basic Routing**

  Pada dasarnya Routing di Laravel membutuhkan informasi mengenai http verb kemudian input berupa url dan apa yang harus dilakukan ketika menerima url tersebut. Untuk membuat sebuah route anda dapat menggunakan callback function atau menggunakan sebuah controller.

Langkah-langkah Praktikum:

a. Pada bagian ini, kita akan membuat dua buah route dengan ketentuan sebagai berikut.

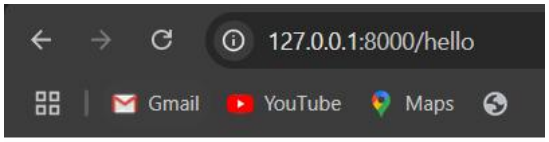
No	Http Verb	Url	Fungsi
1	get	/hello	Tampilkan String Hello ke browser.
2	get	/world	Tampilkan String World ke browser

Kita akan menggunakan project minggu sebelumnya yaitu PWL_2024.

b. Buka file `routes/web.php`. Tambahkan sebuah route untuk nomor 1 seperti di bawah ini:

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

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

c. Buka browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL_2024/public/hello. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.

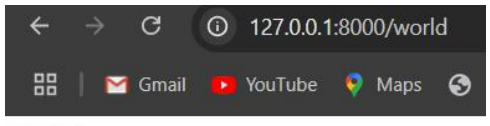


Hello World

d. Untuk membuat route kedua, tambahkan route /world seperti di bawah ini:

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

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

e. Bukalah pada browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL_2024/public/world. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.



World

f. Selanjutnya, cobalah membuat route '/' yang menampilkan pesan 'Selamat Datang'.
g. Kemudian buatlah route '/about' yang akan menampilkan NIM dan nama Anda.

- **Route Parameters**

  Terkadang saat membuat sebuah URL, kita perlu mengambil sebuah parameter yang merupakan bagian dari segmen URL dalam route kita. Misalnya, kita membutuhkan nama user yang dikirim melalui sebuah URL.

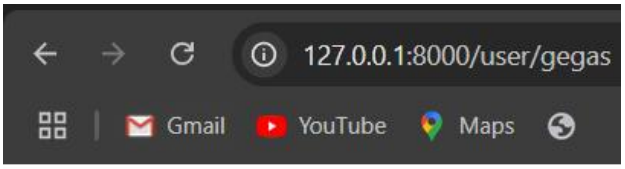
Langkah-langkah Praktikum:

Untuk membuat routing dengan parameter dapat dilakukan dengan cara berikut ini.

a. Kita akan memanggil route */user/{name}* sekaligus mengirimkan parameter berupa nama user $name seperti kode di bawah ini.

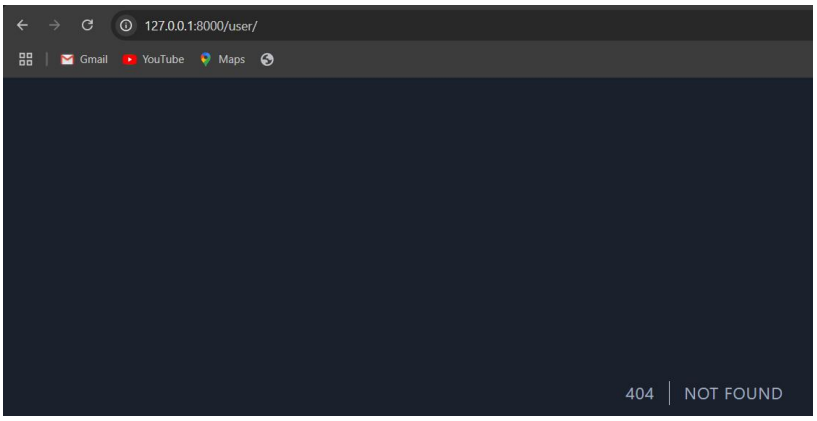
```
Route::get('/user/{name}', function ($name)
    { return 'Nama saya '.$name;
});
```

b. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/user/NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Nama saya gegas

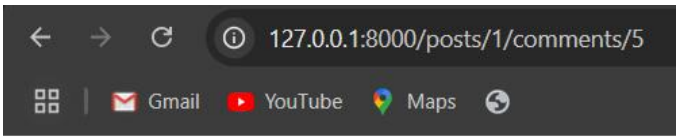
c. Selanjutnya, coba tuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



d. Suatu route, juga bisa menerima lebih dari 1 parameter seperti kode berikut ini. Route menerima parameter $postId dan juga $comment.

```
Route::get('/posts/{post}/comments/{comment}', function
($postId, $commentId) {
    return 'Pos ke-'.$postId." Komentar ke-: ".$commentId;
});
```

e. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/posts/1/comments/5**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Pos ke-1 Komentar ke-: 5

f. Kemudian buatlah route /articles/{id} yang akan menampilkan output "Halaman Artikel dengan ID {id}", ganti id sesuai dengan input dari url.

- **Optional Parameters**

Kita dapat menentukan nilai parameter route, tetapi menjadikan nilai parameter route tersebut opsional. Pastikan untuk memberikan variabel yang sesuai pada route sebagai nilai default. Parameter opsional diberikan tanda '?'.

Langkah-langkah Praktikum:

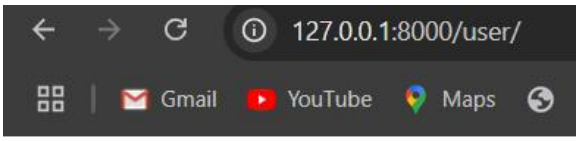
Untuk membuat routing dengan optional parameter dapat dilakukan dengan cara berikut ini.

a. Kita akan memanggil route */user* sekaligus mengirimkan parameter berupa nama user $name dimana parameternya bersifat opsional.

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

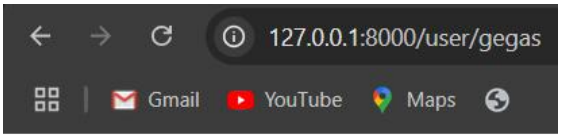
```
    return 'Nama saya '.$name;
});
```

b. Jalankan kode dengan menuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Nama saya

c. Selanjutnya tuliskan URL: **localhost/PWL_2024/public/user/NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

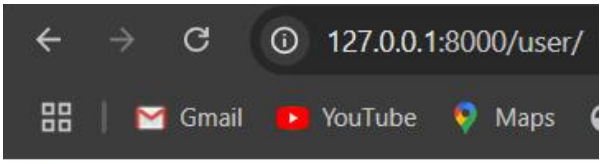


Nama saya gegas

d. Ubah kode pada route /user menjadi seperti di bawah ini.

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

e. Jalankan kode dengan menuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Nama saya John

- **Route Name**

  Route name biasanya digunakan untuk mempermudah kita dalam pemanggilan route saat membangun aplikasi. Kita cukup memanggil name dari route tersebut.

```
Route::get('/user/profile', function () {
    //
})->name('profile');

Route::get(
    '/user/profile',
    [UserProfileController::class, 'show']
)->name('profile');

// Generating URLs...
$url = route('profile');

// Generating Redirects...
return redirect()->route('profile');
```

- **Route Group dan Route Prefixes**

  Beberapa route yang memiliki atribut yang sama seperti middleware yang sama dapat dikelompokkan menjadi satu kelompok untuk mempermudah penulisan route selain

digunakan untuk middleware masih ada lagi penggunaan route group untuk route yang berada dibawah satu subdomain. Contoh penggunaan route group adalah sebagai berikut:

```
Route::middleware(['first', 'second'])->group(function ()
    { Route::get('/', function () {
        // Uses first & second middleware...
    });

Route::get('/user/profile', function () {
        // Uses first & second middleware...
    });
});

Route::domain('{account}.example.com')->group(function ()
    { Route::get('user/{id}', function ($account, $id) {
        //
    });
});

Route::middleware('auth')->group(function ()
     { Route::get('/user', [UserController::class, 'index']);
     Route::get('/post', [PostController::class, 'index']);
     Route::get('/event', [EventController::class, 'index']);

});
```

Route Prefixes

Pengelompokan route juga dapat dilakukan untuk route yang memiliki prefix (awalan) yang sama. Untuk pembuatan route dengan prefix dapat dilihat kode seperti di bawah ini

```
Route::prefix('admin')->group(function ()
     { Route::get('/user', [UserController::class, 'index']);
     Route::get('/post', [PostController::class, 'index']);
     Route::get('/event', [EventController::class, 'index']);

});
```

- **Redirect Routes**

Untuk melakukan redirect pada laravel dapat dilakukan dengan menggunakan Route::redirect cara penggunaannya dapat dilihat pada kode program dibawah ini.

```
Route::redirect('/here', '/there');
```

Redirect ini akan sering digunakan pada kasus kasus CRUD atau kasus lain yang membutuhkan redirect.

- **View Routes**

  Laravel juga menyediakan sebuah route khusus yang memudahkan dalam membuat sebuah routes tanpa menggunakan controller atau callback function. Routes ini langsung menerima input berupa url dan mengembalikan view / tampilan. Berikut ini cara membuat view routes.

```
Route::view('/welcome', 'welcome');
Route::view('/welcome', 'welcome', ['name' => 'Taylor']);
```

  Pada view routes diatas /welcome akan menampilkan view welcome dan pada route kedua /welcome akan menampilkan view welcome dengan tambahan data berupa variabel name.

Simpan perubahan yang telah dilakukan pada Git.

3. Controller

Controller digunakan untuk mengorganisasi logika aplikasi menjadi lebih terstruktur. Logika action aplikasi yang masih ada kaitan dapat dikumpulkan dalam satu kelas Controller. Atau sebuah Controller dapat juga hanya berisi satu buah action. Controller pada Laravel disimpan dalam folder `app/Http/Controllers`.

- **Membuat Controller**

Langkah-langkah Praktikum:

a. Untuk membuat controller pada Laravel telah disediakan perintah untuk menggenerate struktur dasarnya. Kita dapat menggunakan perintah artisan diikuti dengan definisi nama controller yang akan dibuat.

```
php artisan make:controller WelcomeController
```

b. Buka file pada `app/Http/Controllers/WelcomeController.php`. Struktur pada controller dapat digambarkan sebagai berikut:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

```

```
class WelcomeController extends Controller
{
     //
}
```

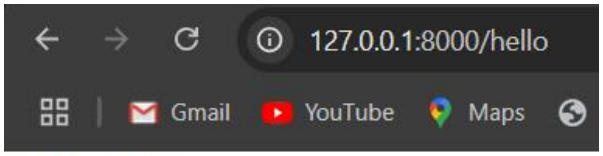
c. Untuk mendefinisikan action, silahkan tambahkan function dengan access public. Sehingga controller di atas menjadi sebagai berikut:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
     public function hello() {
               return 'Hello World';
     }
}
```

d. Setelah sebuah controller telah didefinisikan action, kita dapat menambahkan controller tersebut pada route. Ubah route /hello menjadi seperti berikut:

```
Route::get('/hello', [WelcomeController::class,'hello']);
```

e. Buka browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL_2024/public/hello. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Hello World

f. Modifikasi hasil pada praktikum poin 2 (Routing) dengan konsep controller. Pindahkan logika eksekusi ke dalam controller dengan nama PageController.

Resource	POST	GET	PUT	DELETE
/		Tampilkan Pesan 'Selamat Datang' `PageController : index`		
/about		Tampilkan Nama dan NIM `PageController : about`		
/articles/ {id}		Tampilkan halaman dinamis 'Halaman Artikel dengan Id {id}' id diganti sesuai input dari url		

		PageController : articles		

g. Modifikasi kembali implementasi sebelumnya dengan konsep Single Action Controller. Sehingga untuk hasil akhir yang didapatkan akan ada `HomeController`, `AboutController` dan `ArticleController`. Modifikasi juga route yang digunakan.

- **Resource Controller**

Khusus untuk controller yang terhubung dengan Eloquent model dan dapat dilakukan operasi CRUD terhadap model Eloquent tersebut, kita dapat membuat sebuah controller yang bertipe Resource Controller. Dengan membuat sebuah resource controller, maka controller tersebut telah dilengkapi dengan method-method yang mendukung proses CRUD, serta terdapat sebuah route resource yang menampung route untuk controller tersebut.

Langkah-langkah Praktikum:

a. Untuk membuatnya dilakukan dengan menjalankan perintah berikut ini di terminal.

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

Perintah ini akan men generate sebuah controller dengan nama PhotoController yang berisi method method standar untuk proses CRUD.

b. Setelah controller berhasil degenerate, selanjutnya harus dibuatkan route agar dapat terhubung dengan frontend. Tambahkan kode program berikut pada file web.php.

```
use App\Http\Controllers\PhotoController;

Route::resource('photos', PhotoController::class);
```

c. Jalankan cek list route (php artisan route:list) akan dihasilkan route berikut ini.

Domain	Method	URI	Name	Action	Middleware
	GET|HEAD	/		Closure	web
	GET|HEAD	api/user		Closure	api
					auth:api
	GET|HEAD	photos	photos.index	App\Http\Controllers\PhotoController@index	web
	POST	photos	photos.store	App\Http\Controllers\PhotoController@store	web
	GET|HEAD	photos/create	photos.create	App\Http\Controllers\PhotoController@create	web
	GET|HEAD	photos/{photo}	photos.show	App\Http\Controllers\PhotoController@show	web
	PUT|PATCH	photos/{photo}	photos.update	App\Http\Controllers\PhotoController@update	web
	DELETE	photos/{photo}	photos.destroy	App\Http\Controllers\PhotoController@destroy	web
	GET|HEAD	photos/{photo}/edit	photos.edit	App\Http\Controllers\PhotoController@edit	web
	GET|HEAD|POST|PUT|PATCH|DELETE|OPTIONS	specialMahasiswa		Closure	web
	GET|POST|HEAD	specialUrl		Closure	web

d. Pada route list semua route yang berhubungan untuk crud photo sudah di generate oleh laravel. Jika tidak semua route pada resource controller dibutuhkan dapat dikurangi dengan mengupdate route pada web.php menjadi seperti berikut ini.

```
Route::resource('photos', PhotoController::class)-
    >only([ 'index', 'show'
]);

Route::resource('photos', PhotoController::class)-
    >except([ 'create', 'store', 'update', 'destroy'
]);
```

Simpan perubahan yang telah dilakukan pada Git.

4. View

Dalam kerangka kerja Laravel, View merujuk pada bagian dari aplikasi web yang bertanggung jawab untuk menampilkan antarmuka pengguna kepada pengguna akhir. View pada dasarnya adalah file template yang digunakan untuk menghasilkan HTML yang akan ditampilkan kepada pengguna.

Blade merupakan templating engine bawaan Laravel. Berguna untuk mempermudah dalam menulis kode tampilan. Dan juga memberikan fitur tambahan untuk memanipulasi data di view yang dilempar dari controller. Blade juga memungkinkan penggunaan plain PHP pada kode View. Karena Laravel menggunakan *templating engine* bawaan Blade, maka setiap *file* View diakhiri dengan .blade.php. Misal: index.blade.php, home.blade.php, product.blade.php.

- **Membuat View**

Langkah-langkah Praktikum:

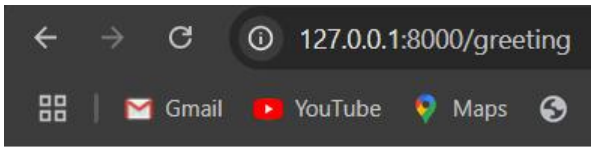
a. Pada direktori app/resources/views, buatlah file hello.blade.php.

```
<!-- View pada resources/views/hello.blade.php -->
<html>
     <body>
     <h1>Hello, {{ $name }}</h1>
     </body>
</html>
```

b. View tersebut dapat dijalankan melalui Routing, dimana *route* akan memanggil View sesuai dengan nama *file* tanpa ‘blade.php’. (Catatan: Gantilah Andi dengan nama Anda)

```
Route::get('/greeting', function () {
     return view('hello', ['name' => 'Andi']);
});
```

c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



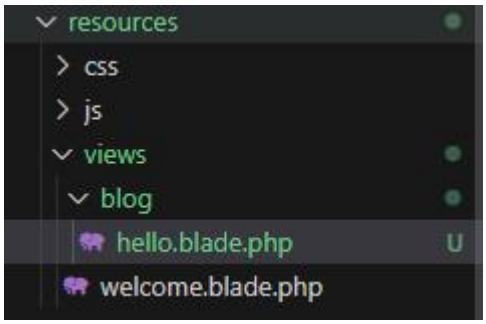
Hello, Gegas

- **View dalam direktori**

Jika di dalam direktori `resources/views` terdapat direktori lagi untuk menyimpan *file* view, sebagai contoh hello.blade.php ada di dalam direktori blog, maka kita bisa menggunakan "dot" notation untuk mereferensikan direktori,

Langkah-langkah Praktikum:

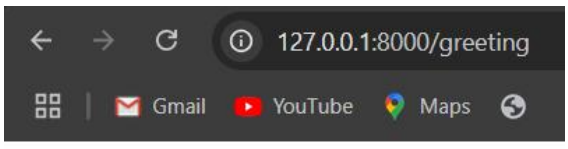
a. Buatlah direktori blog di dalam direktori views.
b. Pindahkan file hello.blade.php ke dalam direktori blog.



c. Selanjutnya lakukan perubahan pada route.

```
Route::get('/greeting', function () {
    return view('blog.hello', ['name' => 'Andi']);
});
```

d. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Hello, Gegas

- **Menampilkan View dari Controller**

View dapat dipanggil melalui Controller. Sehingga Routing akan memanggil Controller terlebih dahulu, dan Controller akan me-*return* view yang dimaksud.

Langkah-langkah Praktikum:

a. Buka WelcomeController.php dan tambahkan fungsi baru yaitu greeting.

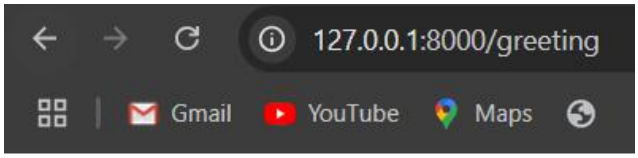
```
class WelcomeController extends Controller
{
    public function
        hello(){ return('Hello
        World');
```

```
    }

    public function greeting(){
        return view('blog.hello', ['name' => 'Andi']);
    }
}
```

b. Ubah route /greeting dan arahkan ke WelcomeController pada fungsi greeting.

```
Route::get('/greeting', [WelcomeController::class,
'greeting']);
```

c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Hello, Andi

- **Meneruskan data ke view**

Pada contoh sebelumnya, kita dapat meneruskan data array ke view agar data tersebut tersedia untuk view:

```
return view('blog.hello', ['name' => 'Andi']);
```

Saat meneruskan informasi dengan cara ini, data harus berupa array dengan pasangan kunci / nilai. Setelah memberikan data ke view, kemudian kita dapat mengakses setiap nilai dalam view menggunakan kunci data seperti: `<?php echo $name; ?>` atau `{{$name}}`. Sebagai alternatif untuk meneruskan array data lengkap ke fungsi view helper, kita dapat menggunakan metode **with** untuk menambahkan bagian data individual ke view. Metode **with** mengembalikan instance view objek sehingga kita dapat melanjutkan rangkaian metode sebelum mengembalikan tampilan

notation untuk mereferensikan direktori,

Langkah-langkah Praktikum:

a. Buka WelcomeController.php dan tambahkan ubah fungsi greeting.

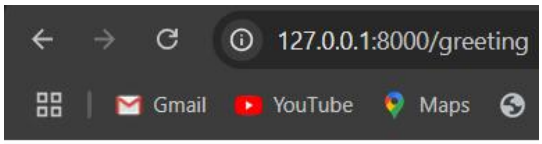
```
class WelcomeController extends Controller
{
    public function
        hello(){ return('Hello
        World');
    }

    public function
        greeting(){ return
        view('blog.hello')
```

```
            ->with('name','Andi')
            ->with('occupation','Astronaut');
    }
}
```

b. Ubah hello.blade.php agar dapat menampilkan dua parameter.

```
<html>
    <body>
    <h1>Hello, {{ $name }}</h1>
    <h1>You are {{ $occupation }}</h1>
    </body>
</html>
```

c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Hello, Gegas

You are Astronaut

Simpan perubahan yang telah dilakukan pada Git.

SOAL PRAKTIKUM

1. Jalankan Langkah-langkah Praktikum pada jobsheet di atas. Lakukan sinkronisasi perubahan pada project PWL_2024 ke Github.
2. Buatlah project baru dengan nama POS. Project ini merupakan sebuah aplikasi Point of Sales yang digunakan untuk membantu penjualan.
3. Buatlah beberapa route, controller, dan view sesuai dengan ketentuan sebagai berikut.

1	Halaman Home Menampilkan halaman awal website
2	Halaman Products Menampilkan daftar product (route prefix) /category/food-beverage /category/beauty-health /category/home-care /category/baby-kid
3	Halaman User Menampilkan profil pengguna (route param) /user/{id}/name/{name}

Route:

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\UserController;
use App\Http\Controllers\ProductController;
use App\Http\Controllers\SalesController;

// Halaman home
Route::get(uri: '/', action: [HomeController::class, 'index']);

// Halaman product
Route::prefix(prefix: 'category')->group(callback: function (): void {
    Route::get(uri: '/food-beverage', action: [ProductController::class, 'foodBeverage']);
    Route::get(uri: '/beauty-health', action: [ProductController::class, 'beautyHealth']);
    Route::get(uri: '/home-care', action: [ProductController::class, 'homeCare']);
    Route::get(uri: '/baby-kid', action: [ProductController::class, 'babyKid']);
});

// Halaman user
Route::get(uri: '/user/{id}/name/{name}', action: [UserController::class, 'show']);

// Halaman Sales
Route::get(uri: '/sales', action: [SalesController::class, 'index']);
```

Controller:

Home Controller

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

2 references | 0 implementations
class HomeController extends Controller
{
    1 reference | 0 overrides
    public function index(): Factory|View {
        return view(view: 'home');
    }
}
```

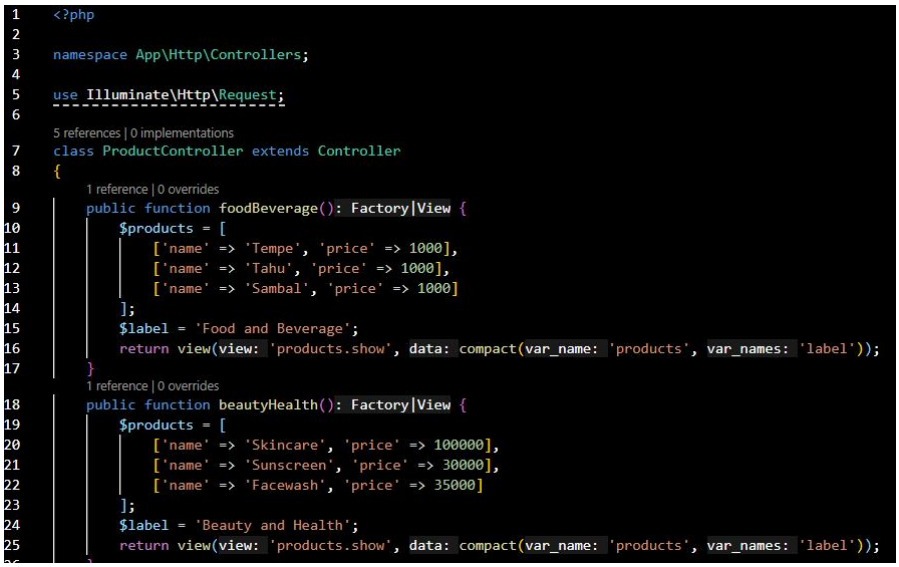
Memiliki method index yang mengembalikan view home.

User Controller

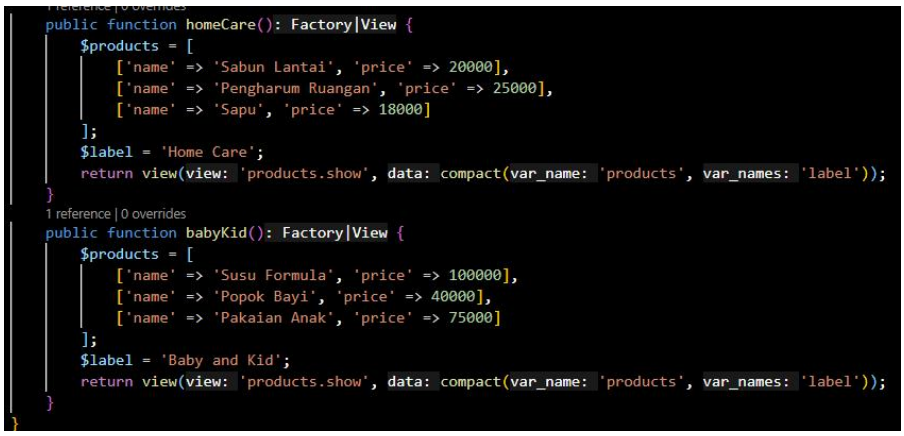
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

2 references | 0 implementations
class UserController extends Controller
{
    1 reference | 0 overrides
    public function show($id, $name): Factory|View {
        return view(view: 'user.profile', data: compact(var_name: 'id', var_names: 'name'));
    }
}
```

Memiliki method show yang menangkap parameter id dan nama yang kemudian mengembalikan view user.profil dan mengirimkan data id dan nama yang telah di tangkap.

Product Controller



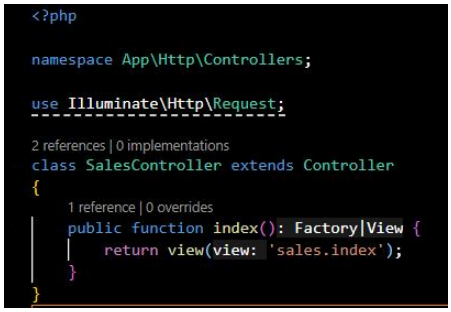
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

5 references | 0 implementations
class ProductController extends Controller
{
    1 reference | 0 overrides
    public function foodBeverage(): Factory|View {
        $products = [
            ['name' => 'Tempe', 'price' => 1000],
            ['name' => 'Tahu', 'price' => 1000],
            ['name' => 'Sambal', 'price' => 1000]
        ];
        $label = 'Food and Beverage';
        return view(view: 'products.show', data: compact(var_name: 'products', var_names: 'label'));
    }
    1 reference | 0 overrides
    public function beautyHealth(): Factory|View {
        $products = [
            ['name' => 'Skincare', 'price' => 100000],
            ['name' => 'Sunscreen', 'price' => 30000],
            ['name' => 'Facewash', 'price' => 35000]
        ];
        $label = 'Beauty and Health';
        return view(view: 'products.show', data: compact(var_name: 'products', var_names: 'label'));
    }
```



```php
    public function homeCare(): Factory|View {
        $products = [
            ['name' => 'Sabun Lantai', 'price' => 20000],
            ['name' => 'Pengharum Ruangan', 'price' => 25000],
            ['name' => 'Sapu', 'price' => 18000]
        ];
        $label = 'Home Care';
        return view(view: 'products.show', data: compact(var_name: 'products', var_names: 'label'));
    }
    1 reference | 0 overrides
    public function babyKid(): Factory|View {
        $products = [
            ['name' => 'Susu Formula', 'price' => 100000],
            ['name' => 'Popok Bayi', 'price' => 40000],
            ['name' => 'Pakaian Anak', 'price' => 75000]
        ];
        $label = 'Baby and Kid';
        return view(view: 'products.show', data: compact(var_name: 'products', var_names: 'label'));
    }
}
```

memiliki 4 method yang masing masing memiliki variabel produk yang berisi data produk dan variabel label yang berisi label kategori dari product tersebut. Kemudian akan di mengembalkan view products.show serta mengirimkan data produk dan label.

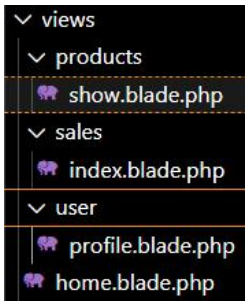
Sales Controller



```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

2 references | 0 implementations
class SalesController extends Controller
{
    1 reference | 0 overrides
    public function index(): Factory|View {
        return view(view: 'sales.index');
    }
}
```

Memiliki method index yang mengembalikan view sales.index

View:



Home view

```blade
<!-- resources/views/show.blade.php -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h1 class="text-center">Selamat Datang di Halaman Home</h1>

    </div>
</body>
</html>
```

menampilkan tampilan selamat datang.

Profile view

```blade
<!-- resources/views/user/profile.blade.php -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Profile</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h1 class="text-center">User Profile</h1>

        <div class="card mx-auto" style="width: 50%;">
            <div class="card-body">
                <h5 class="card-subtitle mb-2 text-muted">User ID: {{ $id }}</h5>
                <h5 class="card-subtitle mb-2 text-muted">Name: {{ $name }}</h5>
            </div>
        </div>

    </div>
</body>
</html>
```

menampilkan data id dan nama yang telah dikirimkan pada method show pada user controller.

Product view

```blade
<!-- resources/views/products/show.blade.php -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product</title>
    <!-- Menggunakan Bootstrap untuk tampilan yang lebih menarik -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h1 class="text-center">{{ $label }}</h1>
        <table class="table table-bordered mt-4">
            <thead class="table-dark">
                <tr>
                    <th>No</th>
                    <th>Product Name</th>
                    <th>Price</th>
                </tr>
            </thead>
            <tbody>
                @foreach($products as $index => $product)
                    <tr>
                        <td>{{ $index + 1 }}</td>
                        <td>{{ $product['name'] }}</td>
                        <td>Rp {{ number_format(num: $product['price'], decimals: 0, decimal_separator: ',', thousands…'.') }}</td>
                    </tr>
                @endforeach
            </tbody>
        </table>
    </div>
</body>
</html>
```

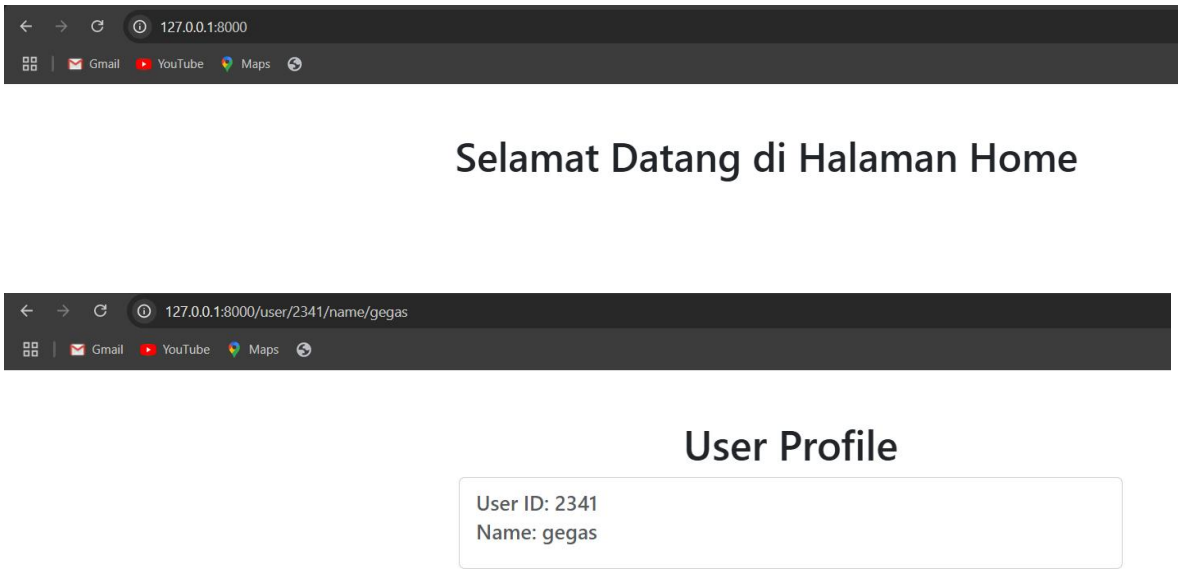
menampilkan data product serta label yang telah dikirimkan dari masing masing method pada product contorller.

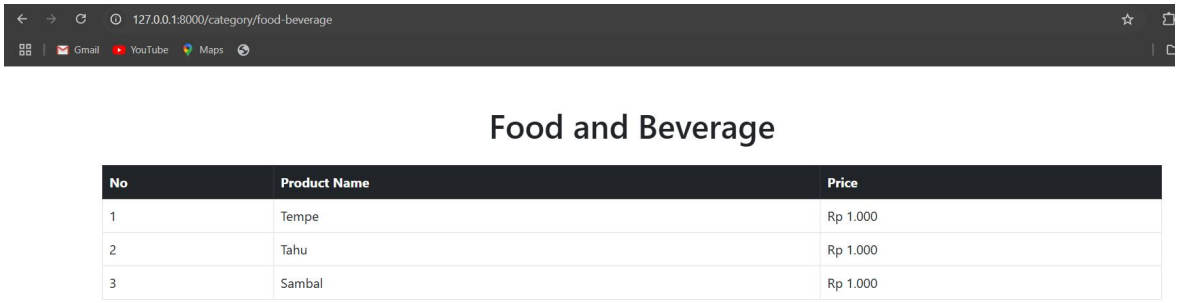
Sales view

```blade
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sales Page</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <div class="container mt-4 text-center">
        <h1 class="display-4">Halaman Penjualan</h1>
        <p>Ini adalah halaman penjualan sederhana.</p>
        <a href="/" class="btn btn-primary">Kembali ke Home</a>
    </div>
</body>
</html>
```

Menampilkan halaman penjualan

Hasil:





No	Product Name	Price
1	Tempe	Rp 1.000
2	Tahu	Rp 1.000
3	Sambal	Rp 1.000

Beauty and Health

No	Product Name	Price
1	Skincare	Rp 100.000
2	Sunscreen	Rp 30.000
3	Facewash	Rp 35.000

Home Care

No	Product Name	Price
1	Sabun Lantai	Rp 20.000
2	Pengharum Ruangan	Rp 25.000
3	Sapu	Rp 18.000

Baby and Kid

No	Product Name	Price
1	Susu Formula	Rp 100.000
2	Popok Bayi	Rp 40.000
3	Pakaian Anak	Rp 75.000

Halaman Penjualan

Ini adalah halaman penjualan sederhana.

4	Halaman Penjualan Menampilkan halaman transaksi POS

4. Route tersebut menjalankan fungsi pada Controller yang berbeda di setiap halaman.
5. Fungsi pada Controller akan memanggil view sesuai halaman yang akan ditampilkan.
6. Simpan setiap perubahan yang dilakukan pada project POS pada Git, sinkronisasi perubahan ke Github.

**** Sekian, dan selamat belajar ****