Nama	**: Dini Elminingtyas**
Kelas/No.absen	**: SIB 2B / 07**
NIM	**: 2341760180**

PEMROGRAMAN WEB LANJUT

JOBSHEET 02

ROUTING, CONTROLLER, DAN VIEW

I. ROUTING

1. Basic Routing

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:

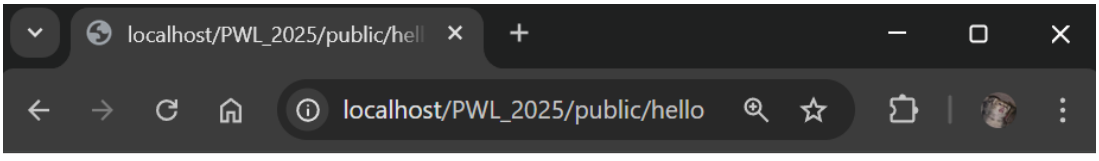
```
use Illuminate\Support\Facades\Route;

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

⇨ Source code;

```
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.

⇨ Halaman browser



Hello World

Berdasarkan hasil yang ditampilkan pada halaman tersebut dapat disimpulkan bahwa penggunaan return mengembalikan atau menampilkan nilai yang kita isikan, dan fungsi get digunakan untuk routing, atau mengarahkan kita sesuai dengan domain yang sudah ditentukan. Pada contoh diatas fungsi get secara langsung mengarahkan ke output Hello World ketika kita mengakses domain /hello

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

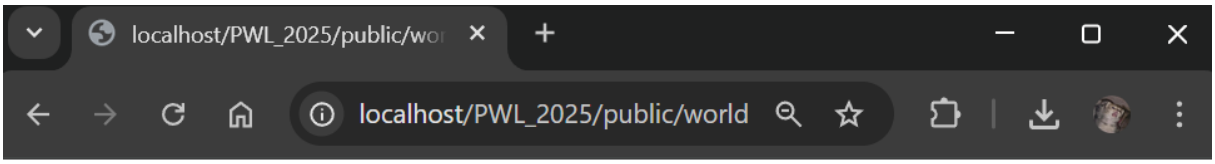
```
use Illuminate\Support\Facades\Route;

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

⇨ Source code

```
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.

⇨ Halaman Browser



World

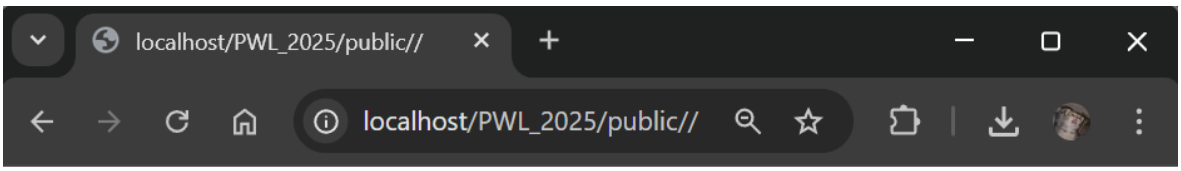
Sama seperti sebelumnya, pada route kedua ini juga menggunakan logic yang sama, fungsi get akan mengarahkan kita ke Output World, ketika mengakses domain / world, sesuai dengan nilai return yang sudah diisikan.

f. Selanjutnya, cobalah membuat route '/' yang menampilkan pesan 'Selamat Datang'.

⇨ Source code

```
Route::get('/', function(){
    return 'Selamat Datang';
});
```

⇨ Halaman browser



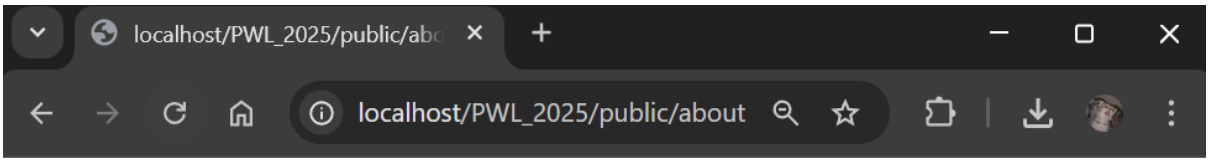
Selamat Datang

g. Kemudian buatlah route '/about' yang akan menampilkan NIM dan nama Anda.

⇨ Source code

```
Route::get('/about', function(){
    return 'Nama : Dini Elminingtyas  ,  NIM : 2341760180' ;
});
```

⇨ Halaman browser



Nama : Dini Elminingtyas , NIM : 2341760180

2. Route Parameters

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

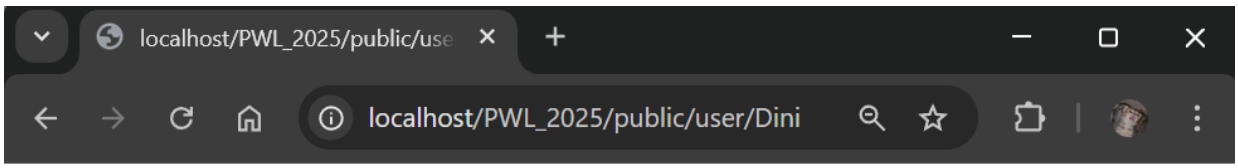
```
Route::get('/user/{name}', function ($name) {
    return 'Nama saya '.$name;
});
```

⇨ Source code

```
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.

⇨ Halaman browser

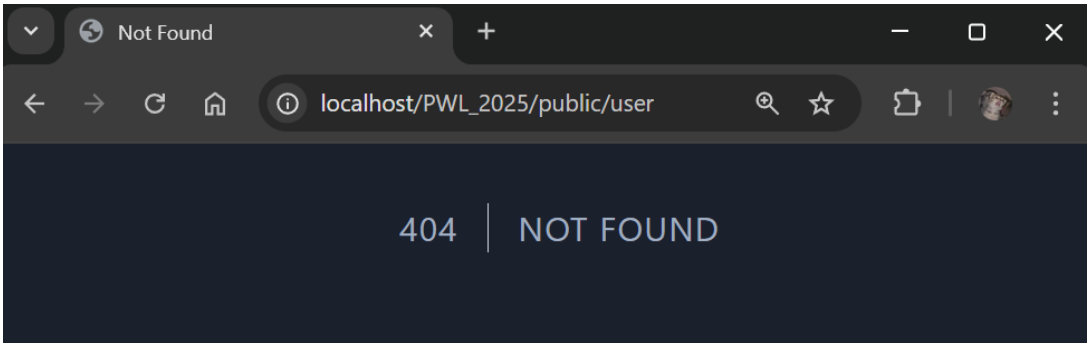


Nama Saya Dini

Untuk routing yang ini, sebenarnya menggunakan logic yang hampir sama dengan yang sebelumnya, tetapi yang membedakan adalah nilai dari return disesuaikan dengan nilai yang dimasukkan ke domain. Seperti yang terlihat pada source code, ketika domain yang diakses adalah '/user/$name' maka akan menempilkan nilai 'Nama Saya $name' yang mana $name ini disesuaikan dengan domain yang diisikan, jadi ketika mengakses domain '/user/Dini' maka akan menampilkan nilai 'Nama Saya Dini'

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

⇨ Halaman browser



Terjadi error, dikarenakan fungsi yang dijalankan untuk route ini menggunakan parameter $name, yang mana tidak bisa mengembalikan nilai jika parameternya null.

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;
});
```

⇨ Source code

```
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.

⇨ Halaman browser

Pos ke-1Komentar ke : 5

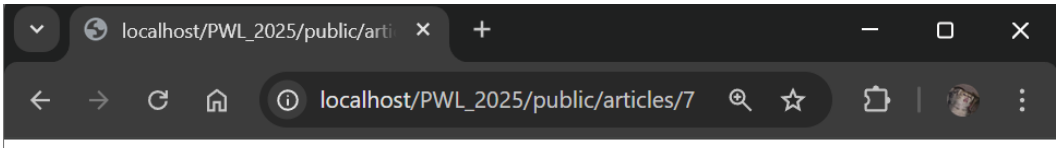
Menggunakan logic yang sama dengan yang sebelunya, tetapi kita bisa menginput lebih dari satu data pada domain tersebut, Seperti pada hasil yang terlihat, kita bisa melakukan input pada posts, dan comments dan hasil yang ditampilkan sesuai dengan input kita pada domain tersebut.

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

⇨ Source code

```
Route::get('/articles/{Id}', function($Id){
    return 'Halaman Artikel Dengan ID : ' . $Id;
});
```

⇨ Halaman browser



Halaman Artikel Dengan ID : 7

3. Optional Parameters

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

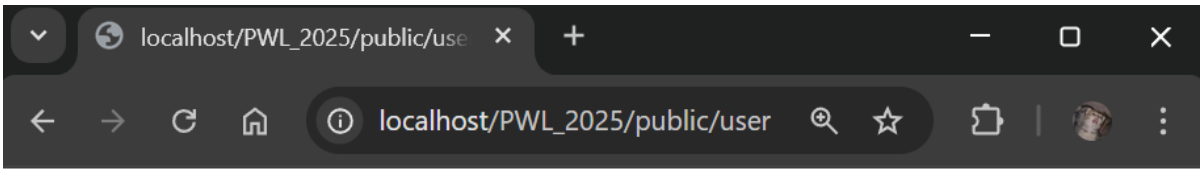
```
Route::get('/user/{name?}', function ($name=null) {
    return 'Nama saya '.$name;
});
```

⇨ Source code

```
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.

⇨ Halaman browser

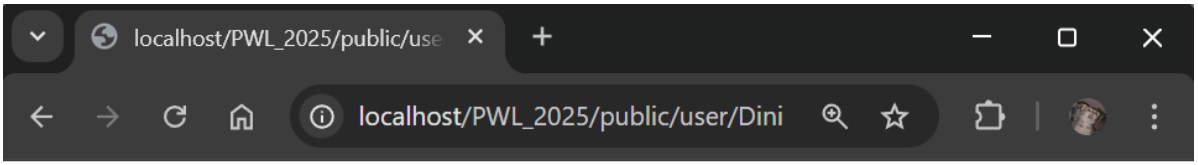


Nama Saya

Dari hasil tersebut dapat disimpulkan bahwa inputan parameter bisa merupakan null atau kosong, sehingga output tetap ditampilkan tanpa adanya input di url atau di domainnya.

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

⇨ Halaman browser



Nama Saya Dini

Seperti pada yang terlihat, kerika parameter diisi maka hasilnya juga menampilkan sesuai dengan input kita pada domain, karena ketika nilai awal parameter adalah null kemudian oarameter tersebut diisi, maka secara otomatis hasilnya juga sesuai dengan input parameter tersebut.

4. 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');
```

5. 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']);

});
```

6. **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.

7. **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.

II. CONTROLLER

1. Membuat Controller

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
```

⇨ Source code

```
PS D:\laragon\www\PWL_2025> php artisan make:controller WelcomeController

  INFO  Controller [D:\laragon\www\PWL_2025\app\Http\Controllers\WelcomeController.php] created successfully.

PS D:\laragon\www\PWL_2025>
```

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
{
     //
}
```

⇨ Source code

app > Http > Controllers > WelcomeController.php > ...

```
1   <?php
2
3   namespace App\Http\Controllers;
4
5   use Illuminate\Http\Request;
6
7   class WelcomeController extends Controller
8   {
9       //
10  }
```

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';
     }
}
```

⇨ Source code

app > Http > Controllers > WelcomeController.php > ...

```
<?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:

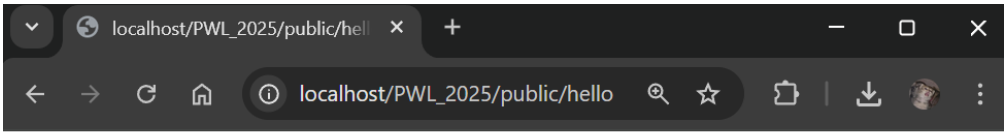
```
Route::get('/hello', [WelcomeController::class,'hello']);
```

⇨ Source code

```
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.

⇨ Halaman browser



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`		

⇨ Source code

PageController.php

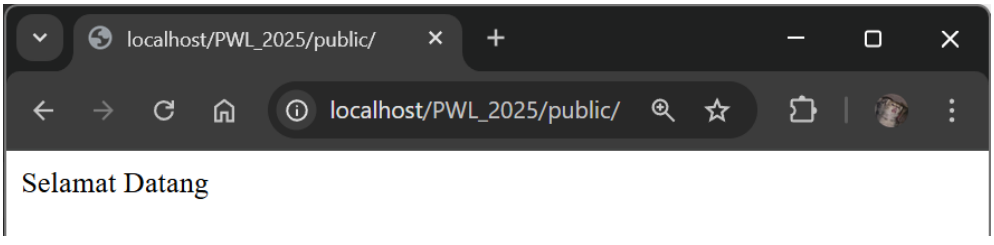
app > Http > Controllers > PageController.php > ...

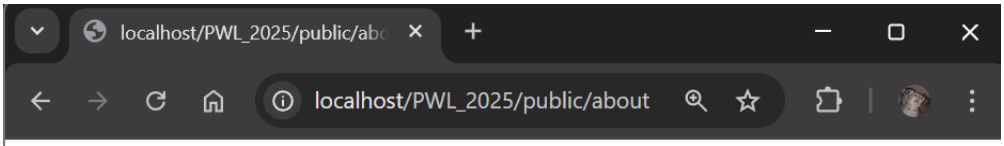
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PageController extends Controller
{
    public function index(){
        return 'Selamat Datang';
    }

    public function about(){
        return 'Nama : Dini Elmiingtyas || NIM : 2341760180';
    }

    public function articles($Id){
        return 'Halaman artikel dengan ID : ' . $Id ;
    }
}
```

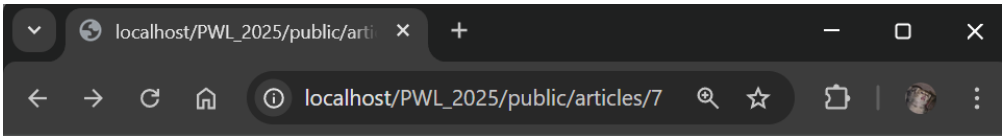
Web.php

```php
Route::get('/', [PageController::class, 'index']);
Route::get('/about', [PageController::class, 'about']);
Route::get('/articles/{id}', [PageController::class, 'articles']);
```

Halaman browser







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.

⇨ Source code

Make the controller

```
PS D:\laragon\www\PWL_2025> php artisan make:controller Homecontroller

  INFO  Controller [D:\laragon\www\PWL_2025\app\Http\Controllers\Homecontroller.php] created successfully.

PS D:\laragon\www\PWL_2025> php artisan make:controller AboutController

  INFO  Controller [D:\laragon\www\PWL_2025\app\Http\Controllers\AboutController.php] created successfully.

PS D:\laragon\www\PWL_2025> php artisan make:controller ArticleController

  INFO  Controller [D:\laragon\www\PWL_2025\app\Http\Controllers\ArticleController.php] created successfully.
```

Single Action Controller

app > Http > Controllers > Homecontroller.php > Homecontroller

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Homecontroller extends Controller
{
    public function index(){
        return 'Selamat Datang';
    }
}

```

app > Http > Controllers > AboutController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class AboutController extends Controller
{
    public function about(){
        return 'Nama : Dini Elminingtyas || NIM : 2341760180';
    }
}
```

app > Http > Controllers > ArticleController.php > ArticleController > articles

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ArticleController extends Controller
{
    public function articles($Id){
        return 'Halaman artikel dengan ID : ' . $Id ;
    }
}
```

Route Modification

```php
use App\Http\Controllers\Homecontroller;
use App\Http\Controllers\AboutController;
use App\Http\Controllers\ArticleController;
```

```php
Route::get('/', [Homecontroller::class, 'index']);
Route::get('/about', [AboutController::class, 'about']);
Route::get('/articles/{id}', [ArticleController::class, 'articles']);
```

2. Resource Controller

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.

⇨ Source code

```
PS D:\laragon\www\PWL_2025> php artisan make:controller PhotoController --resource

  INFO  Controller [D:\laragon\www\PWL_2025\app\Http\Controllers\PhotoController.php] created successfully.
```

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);
```

⇨ Source code

```
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

⇨ Source code

```
PS D:\laragon\www\PWL_2025> php artisan route:list

  GET|HEAD        / ..... Homecontroller@index
  POST            _ignition/execute-solution ..... ignition.executeSolution › Spatie\LaravelIgnition › ExecuteSolutionController
  GET|HEAD        _ignition/health-check ..... ignition.healthCheck › Spatie\LaravelIgnition › HealthCheckController
  POST            _ignition/update-config ..... ignition.updateConfig › Spatie\LaravelIgnition › UpdateConfigController
  GET|HEAD        about ..... AboutController@about
  GET|HEAD        api/user .....
  GET|HEAD        articles/{Id} .....
  GET|HEAD        articles/{id} ..... ArticleController@articles
  GET|HEAD        hello ..... WelcomeController@hello
  GET|HEAD        items ..... items.index › ItemController@index
  POST            items ..... items.store › ItemController@store
  GET|HEAD        items/create ..... items.create › ItemController@create
  GET|HEAD        items/{item} ..... items.show › ItemController@show
  PUT|PATCH       items/{item} ..... items.update › ItemController@update
  DELETE          items/{item} ..... items.destroy › ItemController@destroy
  GET|HEAD        items/{item}/edit ..... items.edit › ItemController@edit
  GET|HEAD        photos ..... photos.index › PhotoController@index
  POST            photos ..... photos.store › PhotoController@store
  GET|HEAD        photos/create ..... photos.create › PhotoController@create
  GET|HEAD        photos/{photo} ..... photos.show › PhotoController@show
  PUT|PATCH       photos/{photo} ..... photos.update › PhotoController@update
  DELETE          photos/{photo} ..... photos.destroy › PhotoController@destroy
  GET|HEAD        photos/{photo}/edit ..... photos.edit › PhotoController@edit
  GET|HEAD        posts/{post}/comments/{comment} .....
  GET|HEAD        sanctum/csrf-cookie ..... sanctum.csrf-cookie › Laravel\Sanctum › CsrfCookieController@show
  GET|HEAD        user/{name?} .....
  GET|HEAD        user/{name} .....
  GET|HEAD        world .....

                                                                                                                  Showing [28] routes
```

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'
]);
```

⇨ Source code

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

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

III. VIEW

1. Membuat view

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>
```

⇨ Source code

resources > views > hello.blade.php > html

```
1   <html>
2       <body>
3           <h1>Hello, {{ $name }}</h1>
4       </body>
5   </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)

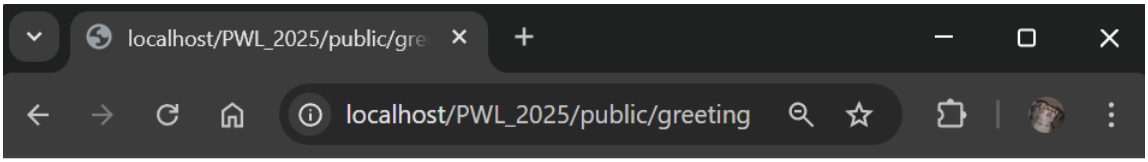
```
Route::get('/greeting', function () {
    return view('hello', ['name' => 'Andi']);
});
```

⇨ Source code

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

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

⇨ Halaman browser



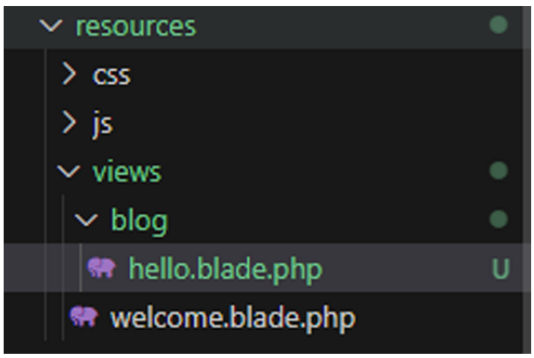
Hello, Dini

Jadi, tampilan diatas dibuat dengan menggabungkan view dan route, yang mana untuk output yang akan ditampilkan dibuat didalam file view yaitu hello.blade.php, yang kemudian akan dipanggil menggunakan route. Nah untuk tampilan tersebut selain telah diatur di file view, juga diatur di dalam route, yang mana dalam route kita mengisikan parameter $name yang mana di file view sebelumnya masih null.

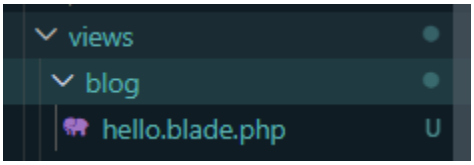
2. View dalam direktori

a. Buatlah direktori blog di dalam direktori views.

b. Pindahkan file hello.blade.php ke dalam direktori blog.



⇨ Buat direktori dan memindahkan hello.blade.php



c. Selanjutnya lakukan perubahan pada route.

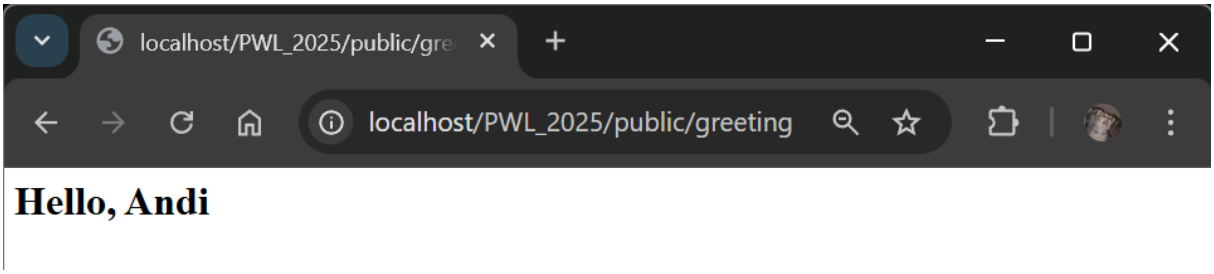
```
Route::get('/greeting', function () {
    return view('blog.hello', ['name' => 'Andi']);
});
```

⇨ Source code

```
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.

⇨ Halaman browser



Meskipun file view sebelumnya telah dipindahkan kedalam direktori baru, tetapi url masih bisa diakses dan menampilkan pesan, hal ini terjadi karena kita telah memodifikasi path pada route, yang mengarahkan ke file view di dalam direktori yang baru.

3. Menampilkan view dari controller

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']);
    }
}
```

⇨ Source code

app > Http > Controllers > WelcomeController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

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.

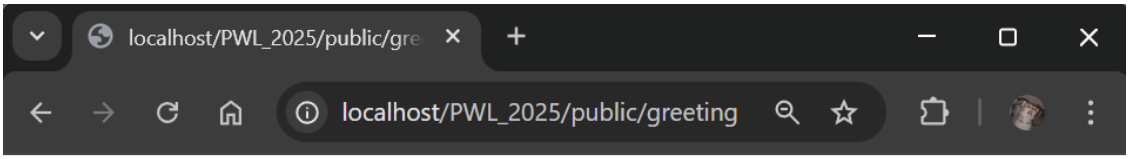
```
Route::get('/greeting', [WelcomeController::class, 'greeting']);
```

⇨ Source code

```php
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.

⇨ Halaman browser



Hello, Andi

Dari percobaan tersebut, dapat dilihat bahwa pemanggilan output dapat dilakukan dengan menggunakan controller. Yang mana dilakukan dengan membuat suatu fungsi, kemudian mengatur nilai return yang nantinya akan ditampilkan, dan memanggilnya melalui route.

4. Meneruskan data ke view

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');
    }
}
```

⇨ Source code

app > Http > Controllers > WelcomeController.php > ...

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

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

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

    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>
```

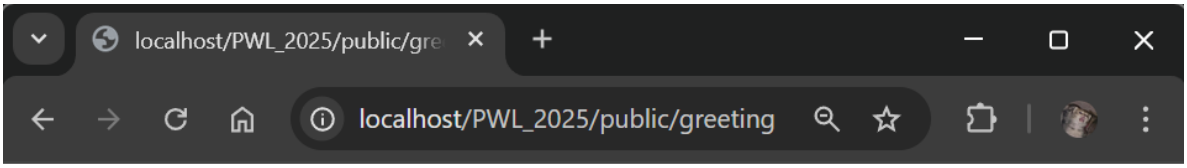
⇨ Source code

resources > views > blog > hello.blade.php > ...

```
<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.

⇨ Halaman browser



Hello, Andi

You are Astronaut

Dari perubahan yang dilakukan, dapat dilihat bahwa kita bisa meneruskan data kedalam view, dengan cara menggunakan with diikuti oleh parameter, dan juga inputannya.

IV. 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.

⇨ Create project POS

```
C:\Users\Dee>composer create-project laravel/laravel="10.3.*" POS
Creating a "laravel/laravel=10.3.*" project at "./POS"
Installing laravel/laravel (v10.3.3)
  - Installing laravel/laravel (v10.3.3): Extracting archive
Created project in C:\Users\Dee\POS
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
  - Locking brick/math (0.12.1)
  - Locking carbonphp/carbon-doctrine-types (2.1.0)
  - Locking dflydev/dot-access-data (v3.0.3)
  - Locking doctrine/inflector (2.0.10)
```

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}
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.

⇨ Source code

a. Homepage

Route

```php
Route::get('/', [HomepageController::class, 'index'])-> name('homepage');
```

Controller

app > Http > Controllers > HomepageController.php > ...

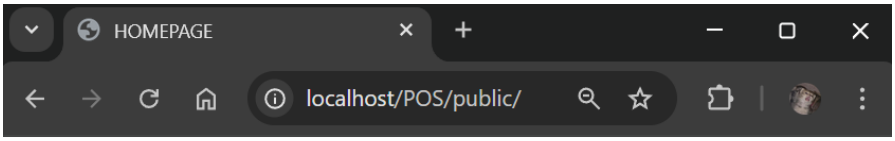
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomepageController extends Controller
{
    public function index(){
        return view('homepage.index');
    }
}
```

View

resources > views > welcome.blade.php > ...

```blade.php
<html>
    <head>
        <title>HOMEPAGE</title>
    </head>
    <body>
        <h1>Welcome To The Hompage</h1>
    </body>
</html>
```

Halaman browser



Welcome To The Hompage

b. Products page

Route

```php
Route::prefix('product')->group(function() {
    Route::get('/', [ProductController::class, 'index'])->name('product.index');
    Route::get('/category/food-beverage', [ProductController::class, 'foodBeverage'])->name('product.food-beverage');
    Route::get('/category/beauty-health', [ProductController::class, 'beautyHealth'])->name('product.beauty-health');
    Route::get('/category/home-care', [ProductController::class, 'homeCare'])->name('product.home-care');
    Route::get('/category/baby-kid', [ProductController::class, 'babyKids'])->name('product.baby-kid');
});
```

Controller

app > Http > Controllers > ProductController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller
{
    public function index(){
        return view('product');
    }

    public function foodBeverage(){
        return view('foodBeverage');
    }

    public function beautyHealth(){
        return view('beautyHealth');
    }

    public function homeCare(){
        return view('homeCare');
    }

    public function babyKids(){
        return view('babyKid');
    }
}
```

View

- Product index

resources > views > product.blade.php > ...

```
<html>
    <head>
        <title>Product Page</title>
    </head>
    <body>
        <h1>Product Index Page</h1>
        <h2>Product Category : </h2>
        <ul>
            <li><a href="{{ route('product.food-beverage') }}">Food & Beverage</a></li>
            <li><a href="{{ route('product.beauty-health') }}">Beauty & Health</a></li>
            <li><a href="{{ route('product.home-care') }}">Home Care</a></li>
            <li><a href="{{ route('product.baby-kid') }}">Baby & Kids</a></li>
        </ul>
    </body>
</html>
```

- Food & Beverage

resources > views > homeCare.blade.php > ...

```
<html>
    <head>
        <title>Home Care</title>
    </head>
    <body>
        <h1>Home Care Products</h1>
    </body>
</html>
```

- Beauty & Health

resources > views > beautyHealth.blade.php > ...

```
<html>
    <head>
        <title>Beauty & Health</title>
    </head>
    <body>
        <h1>Beauty & Health Products</h1>
    </body>
</html>

```

- Home Care

resources > views > homeCare.blade.php > ...

```
<html>
    <head>
        <title>Home Care</title>
    </head>
    <body>
        <h1>Home Care Products</h1>
    </body>
</html>

```

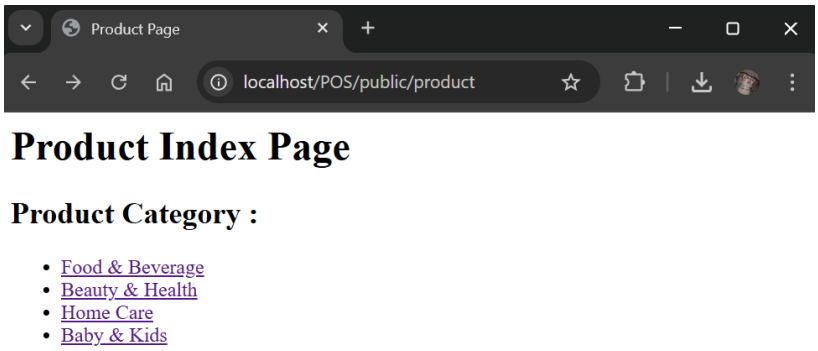
- Baby & Kid

resources > views > babyKid.blade.php > ...

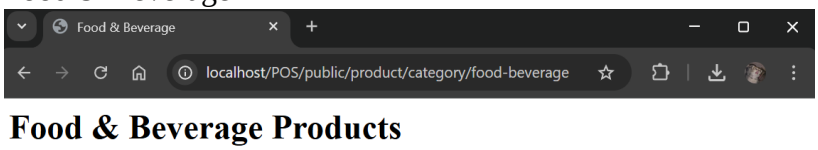
```
<html>
    <head>
        <title>Baby & Kids</title>
    </head>
    <body>
        <h1>Baby & Kids Products</h1>
    </body>
</html>

```

Halaman browser

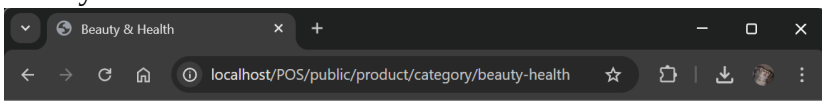
- Product Index



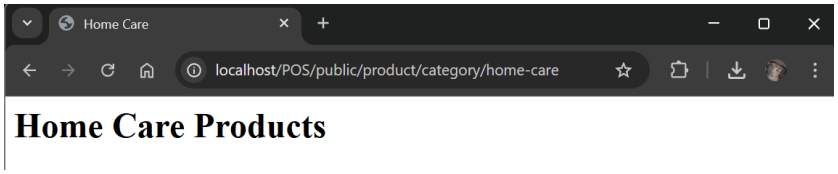
- Food & Beverage



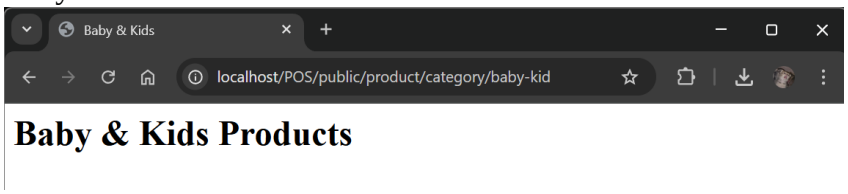
- Beauty & Health



- Home Care



- Baby & Kids



c. User page

Route

```php
Route::get('/user/{id}/name/{name}', [UserController::class, 'show']);
```

Controller

app > Http > Controllers > UserController.php > ...

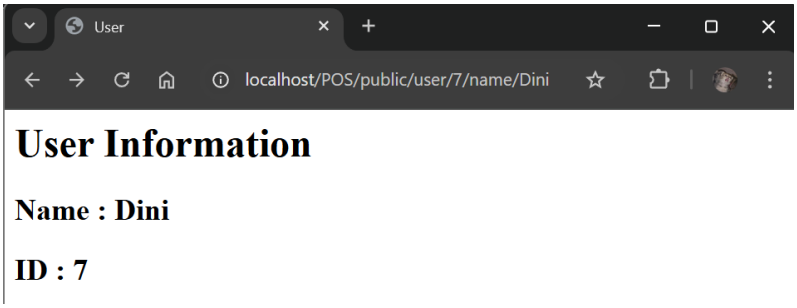
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class UserController extends Controller
{
    public function show($id, $name) {
        return view('user')
            ->with('id', $id)
            ->with('name', $name);
    }
}
```

View

resources > views > user.blade.php > ...

```blade
<html>
    <head>
        <title>User</title>
    </head>
    <body>
        <h1>User Information</h1>
        <h2>Name : {{ $name }}</h2>
        <h2>ID : {{ $id }}</h2>
    </body>
</html>

```

Halaman browser



d. Transaction page

Route

```php
Route::get('/sales', function(){
    return view('transaction');
});
```

Controller

app > Http > Controllers > TransactionController.php > ...

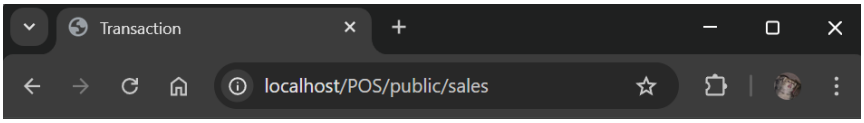
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class TransactionController extends Controller
{
    public function index(){
        return view('transaction');
    }
}
```

View

resources > views > transaction.blade.php > ...

```php
<html>
    <head>
        <title>Transaction</title>
    </head>
    <body>
        <h1>This is Transaction page</h1>
    </body>
</html>
```

Halaman browser



This is Transaction page

e. Link Github project POS
https://github.com/dinielmi/POS