



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

Nama : Nimas Septiandini  
No presensi : 17  
NIM : 2341760087  
Kelas : SIB-2B

## JOBSHEET 02

### ROUTING, CONTROLLER, DAN VIEW

#### 1. MVC pada Laravel

#### 2. Routing

##### - Basic Routing

##### Langkah-langkah Praktikum:

Pada bagian ini, kita akan membuat dua buah route dengan ketentuan sebagai berikut. Kita akan menggunakan project minggu sebelumnya yaitu PWL\_2024.

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

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

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

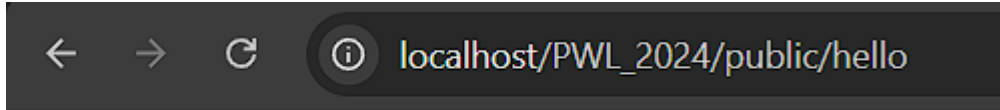
Route::get('/hello', function () {
    return 'Hello World';
});
```

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

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



- b. 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!

= Ya, halaman yang muncul sudah sesuai, halaman menampilkan teks yang diinputkan pada return

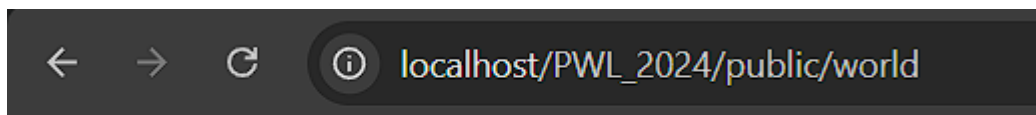
- c. Untuk membuat route kedua, tambahkan route /world seperti di bawah ini:

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

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

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

- d. 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.

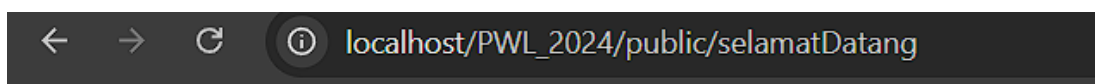


World!

= Ya, sudah sesuai. Dengan memanggil route /world, halaman yang muncul, menampilkan kata 'World!' sesuai value yang terdapat pada return.

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

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

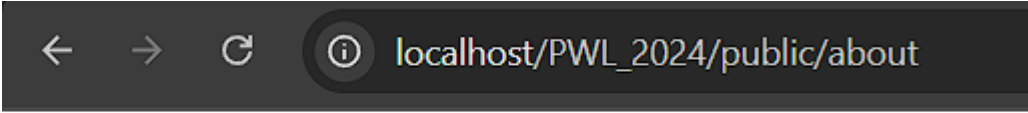


Selamat Datang



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

```
/* Route about */  
Route::get('/about', function() {  
    return '2341760087 Nimas Septiandini';  
});
```



localhost/PWL\_2024/public/about

2341760087 Nimas Septiandini



## - 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:

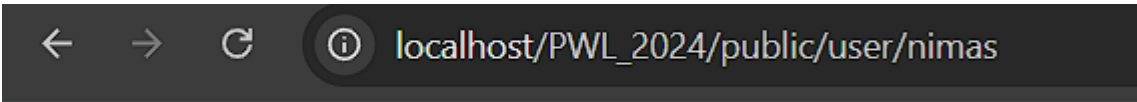
Untuk membuat routing dengan parameter dapat dilakukan dengan cara berikut ini.

- 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;  
});
```

```
/* Jobsheet 2: Route Parameters */  
Route::get('/user/{name}', function ($name) {  
    return 'Nama saya '.$name;  
});
```

- Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL\_2024/public/user>NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

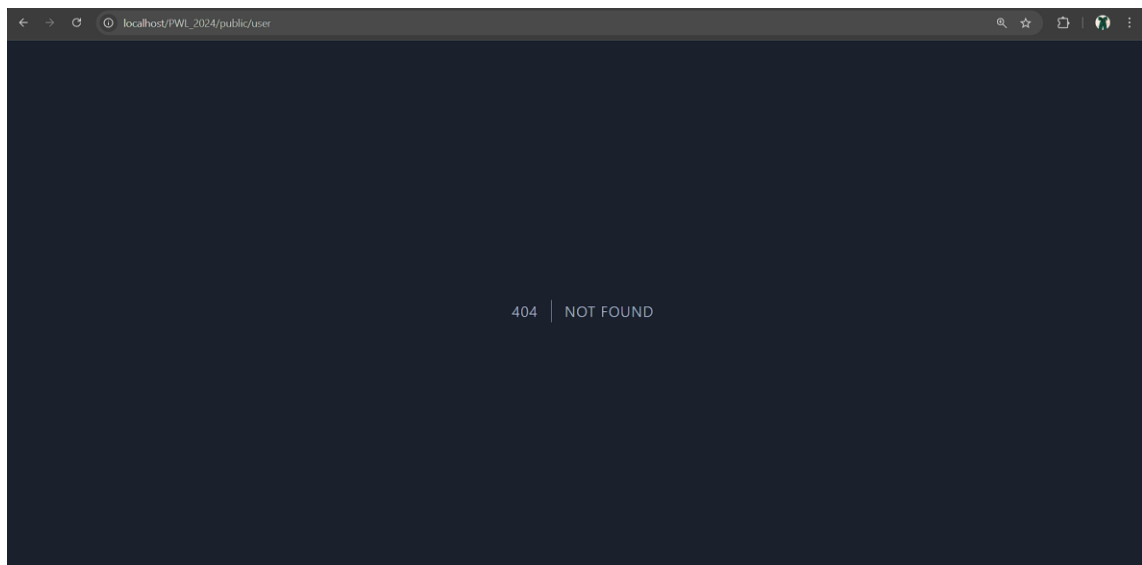


localhost/PWL\_2024/public/user/nimas

Nama saya nimas

= Ketika URL diakses, laravel akan melihat parameter `{name}` dari URL dan meneruskannya ke parameter `$name` yang kemudian parameter tersebut diproses dengan callback untuk ditampilkan sebagai output.

- Selanjutnya, coba tuliskan URL: **localhost/PWL\_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



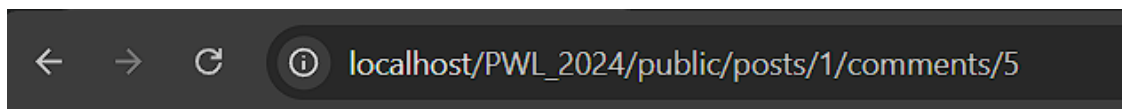
= Halaman tidak ditemukan karena tidak mengakses route apapun

- 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;  
});
```

```
Route::get('/posts/{post}/comments/{comment}', function ($postId, $comment  
    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**

= Route tersebut menerima 2 parameter. Parameter \$postId bernilai 1 dan parameter \$comment bernilai 5 saat URL tersebut diakses

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



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>

```
/*Route id halaman artikel*/  
Route::get('articles/{id}', function ($id) {  
    return 'Halaman Artikel dengan ID '.$id;  
});
```

← → ↻ ⓘ localhost/PWL\_2024/public/articles/17

Halaman Artikel dengan ID 17



## - 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 '?'.

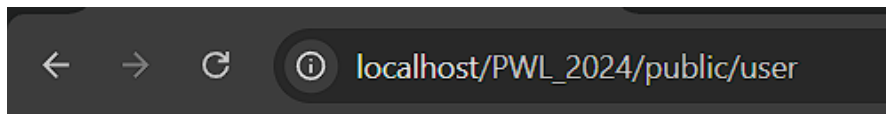
### Langkah-langkah Praktikum:

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;  
});
```

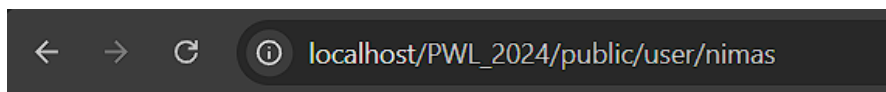
- b. Jalankan kode dengan menuliskan URL: **localhost/PWL\_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



**Nama saya**

= Ketika URL **localhost/PWL\_2024/public/user/** diakses tanpa parameter `{name}`, Laravel menetapkan nilai default `$name = null`, sehingga teks yang ditampilkan hanya **"Nama saya "** tanpa nama. Hal ini terjadi karena `$name` tidak memiliki nilai, sehingga hasil output menjadi tidak lengkap.

- c. Selanjutnya tuliskan URL: **localhost/PWL\_2024/public/user>NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



**Nama saya nimas**

= parameter `{name}` pada URL diisi dengan nama 'nimas' yang kemudian menjadi parameter, melalui fungsi callback nilai dari parameter `$name` ditampilkan sebagai output

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

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



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>

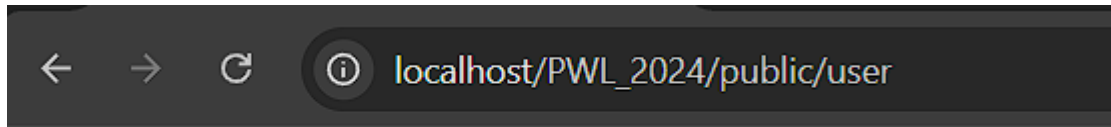
---

```
/* Jobsheet 2: Optional Parameters */  
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.



= Default dari parameter \$name diubah menjadi 'John', sehingga Ketika parameter {name} pada URL tidak diisi, nilai default dari parameter akan ditampilkan.

#### - **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:

- 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
```

```
PS C:\Kuliah\semester3\web\laragon\www\PWL_2024> php artisan make:controller WelcomeController
```

```
INFO Controller [C:\Kuliah\semester3\web\laragon\www\PWL_2024\app\Http\Controllers\WelcomeController.php] created successfully.
```

- 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
{
    //
}
```

```
<?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';
    }
}
```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

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

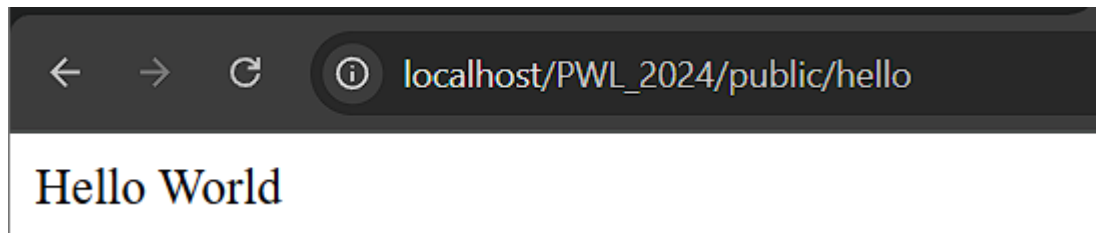
- d. Setelah sebuah controller telah didefinisikan action, kita dapat menambahkan controller tersebut pada route. Ubah route /hello menjadi seperti berikut:

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



```
/* Jobsheet 2: Membuat controller */  
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.



= URL tersebut akan menggunakan route /hello yang kemudian dapat menampilkan hasil return 'Hello World' dari controller WelcomeController.

- 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

```
app > Http > Controllers > PageController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class PageController extends Controller
8  {
9      public function index() {
10         return 'Selamat Datang';
11     }
12
13     public function about() {
14         return '2341760087 Nimas Septiandini';
15     }
16
17     public function articles($id) {
18         return 'Halaman artiker dengan ID ' . $id;
19     }
20 }
21
```

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

localhost/PWL\_2024/public/

Selamat Datang

localhost/PWL\_2024/public/about

2341760087 Nimas Septiandini

localhost/PWL\_2024/public/articles/17

Halaman artiker dengan ID 17

- 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.



```
INFO Controller [C:\Kuliah\semester3\web\laragon\www\PWL_2024\app\Http\Controllers\HomeController.php] created successfully.
```

```
PS C:\Kuliah\semester3\web\laragon\www\PWL_2024> php artisan make:controller AboutController
```

```
INFO Controller [C:\Kuliah\semester3\web\laragon\www\PWL_2024\app\Http\Controllers\AboutController.php] created successfully.
```

```
PS C:\Kuliah\semester3\web\laragon\www\PWL_2024> php artisan make:controller ArticleController
```

```
INFO Controller [C:\Kuliah\semester3\web\laragon\www\PWL_2024\app\Http\Controllers\ArticleController.php] created successfully.
```

HomeController

```
app > Http > Controllers > HomeController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class HomeController extends Controller
8  {
9      public function index() {
10         return 'Selamat Datang';
11     }
12
13 }
```

```
Route::get('/', [HomeController::class, 'index']);
```

localhost/PWL\_2024/public/

# Selamat Datang

AboutController



app > Http > Controllers > AboutController.php

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class AboutController extends Controller
8  {
9      public function about() {
10         return '2341760087 Nimas Septiandini';
11     }
12 }
```

```
Route::get('/about', [AboutController::class, 'about']);
```

localhost/PWL\_2024/public/about

2341760087 Nimas Septiandini

ArticleController

app > Http > Controllers > ArticleController.php

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class ArticleController extends Controller
8  {
9      public function articles($id) {
10         return 'Halaman artiker dengan ID ' . $id;
11     }
12 }
```

```
Route::get('/articles/{id}', [ArticleController::class, 'articles']);
```





← → ↺ ⓘ localhost/PWL\_2024/public/articles/17

## Halaman artiker dengan ID 17

### 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 generate sebuah controller dengan nama PhotoController yang berisi method method standar untuk proses CRUD.

```
PS C:\Kuliah\semester3\web\laragon\www\PWL_2024> php artisan make:controller PhotoController --resource
INFO Controller [C:\Kuliah\semester3\web\laragon\www\PWL_2024\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);
```

```
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
	GET HEAD	photos	photos.index	App\Http\Controllers\PhotoController@index	auth:api
	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



```
PS C:\Kuliah\semester3\web\laragon\www\PWL_2024> php artisan route:list

GET|HEAD / ..... HomeController@index
POST _ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgniti...
GET|HEAD _ignition/health-check ignition.healthCheck > Spatie\LaravelIgnition > Hea...
POST _ignition/update-config ignition.updateConfig > Spatie\LaravelIgnition > U...
GET|HEAD about ..... AboutController@about
GET|HEAD api/user .....
GET|HEAD articles/{id} ..... ArticleController@articles
GET|HEAD hello ..... WelcomeController@hello
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 > CsrfCookieCont...
GET|HEAD selamatDatang .....
GET|HEAD user/{name?} .....
GET|HEAD user/{name} .....
GET|HEAD world .....

Showing [21] 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'
]);
```

**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>
```

```
resources > views > hello.blade.php > ...
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']);  
});
```

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



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

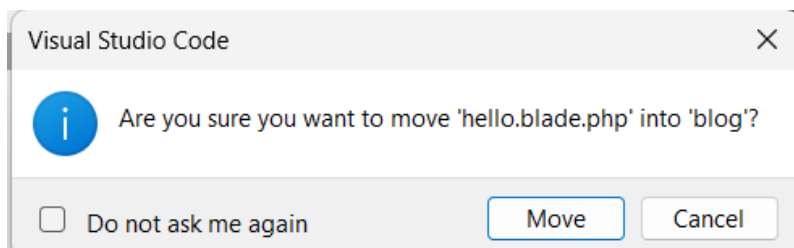
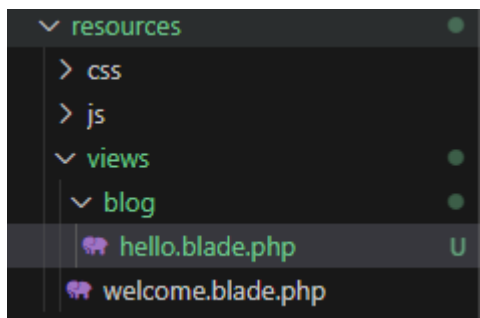


- **View dalam direktori**

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:**

- Buatlah direktori `blog` di dalam direktori `views`.
- Pindahkan file `hello.blade.php` ke dalam direktori `blog`.



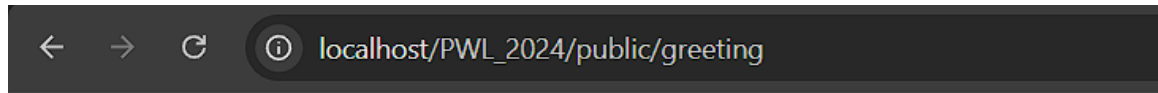
- c. Selanjutnya lakukan perubahan pada route.

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

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



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



# Hello, Nimas Septiandini

= URL akan mengakses route `/greeting` yang akan menampilkan output dari file `hello.blade.php` dengan parameter nama 'Nimas Septiandini'

## - 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']);  
}  
  
}
```

```
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      public function hello() {  
10         return 'Hello World';  
11     }  
12  
13     public function greeting() {  
14         return view('blog.hello', ['name' => 'Nimas Septiandini']);  
15     }  
16 }
```

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

```
Route::get('/greeting', [WelcomeController::class,  
    '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, Nimas Septiandini

## - 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');  
    }  
}
```

```
<?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' => 'Nimas Septiandini']);  
    // }  
  
    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>
```

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

- c. Jalankan code dengan membuka url [localhost/PWL\\_2024/public/greeting](localhost/PWL_2024/public/greeting). Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



← → ↺ ⓘ localhost/PWL\_2024/public/greeting

# Hello, Andi

# You are Astronaut

### Simpan perubahan yang telah dilakukan pada Git.

= Pada WelcomeController, terdapat function greeting yang akan mengakses file hello.blade.php pada view. parameter \$name dan \$occupation mengambil value yang diberikan saat dipanggil di function greeting()

## SOAL PRAKTIKUM

1. Jalankan Langkah-langkah Praktikum pada jobsheet di atas. Lakukan sinkronisasi perubahan pada project PWL\_2024 ke Github.

Github praktikum project: [https://github.com/septyandini921/PWL\\_2024.git](https://github.com/septyandini921/PWL_2024.git)

2. Buatlah project baru dengan nama POS. Project ini merupakan sebuah aplikasi Point of Sales yang digunakan untuk membantu penjualan.

```
PS C:\Kuliah\semester3\web\laragon\www> composer create-project --prefer-dist laravel/laravel POS "10.*"
Creating a "laravel/laravel" project at "./POS"
The repository at "C:\Kuliah\semester3\web\laragon\www" does not have the correct ownership and git refuses to use it:
Fatal: detected dubious ownership in repository at 'C:/Kuliah/semester3/web/laragon/www'
'C:/Kuliah/semester3/web/laragon/www' is owned by:
  BUILTIN\Administrators (S-1-5-32-544)
but the current user is:
  NXST-PLANET/NXST (S-1-5-21-1171658239-3704094650-3913630630-1002)
To add an exception for this directory, call:

    git config --global --add safe.directory C:/Kuliah/semester3/web/laragon/www

Installing laravel/laravel (v10.3.3)
- Installing laravel/laravel (v10.3.3): Extracting archive
Created project in C:\Kuliah\semester3\web\laragon\www\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
---	--



	<pre>app &gt; Http &gt; Controllers &gt; HomeController.php 1  &lt;?php 2 3  namespace App\Http\Controllers; 4 5  use Illuminate\Http\Request; 6 7  class HomeController extends Controller 8  { 9      public function index() 10     { 11         return view('home.index'); 12     } 13 }</pre> <pre>resources &gt; views &gt; home &gt; index.blade.php &gt; html 1  &lt;html&gt; 2  &lt;head&gt; 3      &lt;title&gt;HOMEPAGE&lt;/title&gt; 4  &lt;/head&gt; 5  &lt;body&gt; 6      &lt;h1&gt;Selamat Datang&lt;/h1&gt; 7  &lt;/body&gt; 8  &lt;/html&gt;</pre> <pre>← → ↻ ⓘ 127.0.0.1:8000</pre> <h1>Selamat Datang</h1>
2	<p>Halaman Products</p> <p>Menampilkan daftar product (route prefix)</p> <ul style="list-style-type: none"><li>/category/food-beverage</li><li>/category/beauty-health</li><li>/category/home-care</li><li>/category/baby-kid</li></ul>



```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

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

    public function foodBeverage()
    {
        return view('product.food-beverage');
    }

    public function beautyHealth()
    {
        return view('product.beauty-health');
    }

    public function homeCare()
    {
        return view('product.home-care');
    }

    public function babyKid()
    {
        return view('product.baby-kid');
    }
}
```

```
resources > views > product > baby-kid.blade.php > html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Baby & Kid</title>
5 </head>
6 <body>
7     <h1>Baby & Kid Products</h1>
8 </body>
```

```
resources > views > product > beauty-health.blade.php > html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Beauty & Health</title>
5 </head>
6 <body>
7     <h1>Beauty & Health Products</h1>
8 </body>
9 </html>
```



```
resources > views > product > food-beverage.blade.php > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Food & Beverage</title>
5  </head>
6  <body>
7      <h1>Food & Beverage Products</h1>
8  </body>
9  </html>
```

```
resources > views > product > home-care.blade.php > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Home Care</title>
5  </head>
6  <body>
7      <h1>Home Care Products</h1>
8  </body>
9  </html>
```

```
resources > views > product > index.blade.php > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Product Index</title>
5  </head>
6  <body>
7      <h1>Product Index Page</h1>
8
9      <h2>Kategori Produk:</h2>
10     <ul>
11         <li><a href="{{ route('product.food-beverage') }}">Makanan & Minuman</a></li>
12         <li><a href="{{ route('product.beauty-health') }}">Kecantikan & Kesehatan</a></li>
13         <li><a href="{{ route('product.home-care') }}">Perawatan Rumah</a></li>
14         <li><a href="{{ route('product.baby-kid') }}">Bayi & Anak-anak</a></li>
15     </ul>
16 </body>
17 </html>
```

```
Route::get('/', [HomeController::class, 'index'])->name('home');
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, 'babyKid'])->name('product.baby-kid');
});
```



localhost/POS/resources/views/

## Index of /POS/resources/views

- [Parent Directory](#)
- [home/](#)
- [product/](#)
- [sales/](#)
- [user/](#)
- [welcome.blade.php](#)

localhost/POS/resources/views/product/

## Index of /POS/resources/views/product

- [Parent Directory](#)
- [baby-kid.blade.php](#)
- [beauty-health.blade.php](#)
- [food-beverage.blade.php](#)
- [home-care.blade.php](#)
- [index.blade.php](#)

localhost/POS/resources/views/product/index.blade.php

## Product Index Page

### Kategori Produk:

- [Makanan & Minuman](#)
- [Kecantikan & Kesehatan](#)
- [Perawatan Rumah](#)
- [Bayi & Anak-anak](#)

localhost/POS/resources/views/product/baby-kid.blade.php

## Baby & Kid Products

localhost/POS/resources/views/product/beauty-he...

## Beauty & Health Products

localhost/POS/resources/views/product/food-bev...

## Food & Beverage Products



	<div><div>← → ↺ ⓘ localhost/POS/resources/views/product/home-car... 🔍 ☆ 📁   🗑️ 📄 📑 👤 ⋮</div><h2>Home Care Products</h2></div>
3	<div><div>Halaman User Menampilkan profil pengguna (route param) /user/{id}/name/{name}</div><div><div>app &gt; Http &gt; Controllers &gt; ⓘ UserController.php</div><pre>1 &lt;?php 2 3 namespace App\Http\Controllers; 4 5 use Illuminate\Http\Request; 6 7 class UserController extends Controller 8 { 9     public function show(\$id, \$name) 10    { 11        return view('user.show') 12            -&gt;with('name', \$name) 13            -&gt;with('id', \$id); 14    } 15 }</pre></div><div><div>resources &gt; views &gt; user &gt; ⓘ show.blade.php &gt; 📄 html</div><pre>1 &lt;html&gt; 2 &lt;body&gt; 3     &lt;h1&gt;Hello, Nimas Septiandini &lt;/h1&gt; 4     &lt;h1&gt;ID 17&lt;/h1&gt; 5 &lt;/body&gt; 6 &lt;/html&gt;</pre></div><div><div>← → ↺ ⓘ localhost/POS/resources/views/user/show.blade.php 🔍 ☆ 📁   🗑️ 📄 📑 👤 ⋮</div><h2>Hello, Nimas Septiandini</h2><h2>ID 17</h2></div></div>

4 Halaman Penjualan  
Menampilkan halaman transaksi POS

```
app > Http > Controllers > SalesController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class SalesController extends Controller
8  {
9      public function index()
10     {
11         return view('sales.index');
12     }
13 }
```

```
resources > views > sales > index.blade.php > html
1  <html>
2      <head>
3          <title>Transaction</title>
4      </head>
5      <body>
6          <h1>Ini Halaman Transaksi</h1>
7      </body>
8  </html>
```

localhost/POS/resources/views/sales/index.blade.p...

## Ini Halaman Transaksi

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.

Github: <https://github.com/septyandini921/POS.git>

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