



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 : Lyra Faiqah Bilqis

Kelas : 2A SIB

Absen : 19

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



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>

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;

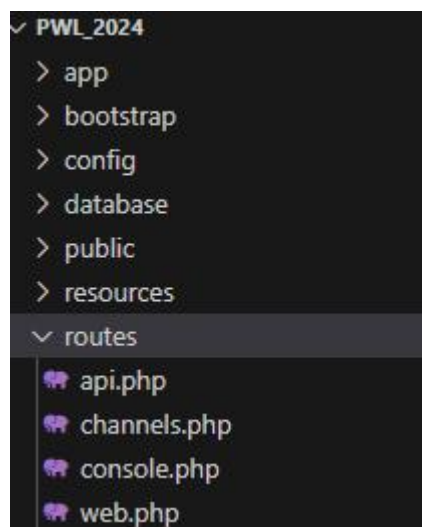
Route::get('/hello', function ()
    { return 'Hello World';
});
```

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:

- 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

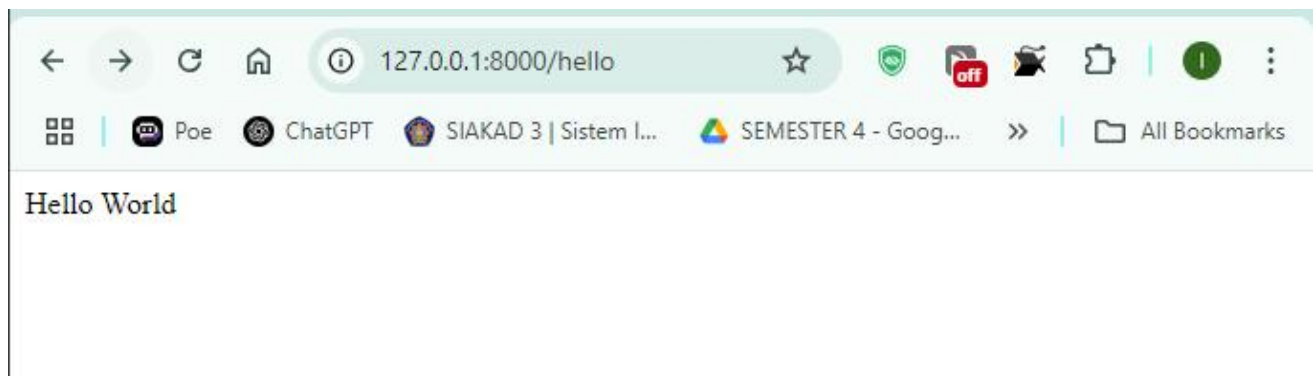
Kita akan menggunakan project minggu sebelumnya yaitu PWL_2024.

- 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. Gambar menunjukkan tampilan halaman web lokal dengan URL `'127.0.0.1:8000/hello'`, yang menampilkan teks "Hello World". Ini menandakan bahwa server web berjalan di komputer lokal dan menangani permintaan ke endpoint `'/hello'`. Kemungkinan besar, aplikasi ini dibuat menggunakan framework seperti Django, Flask, atau Express.js, dengan route yang mengembalikan teks `"Hello World"`.



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

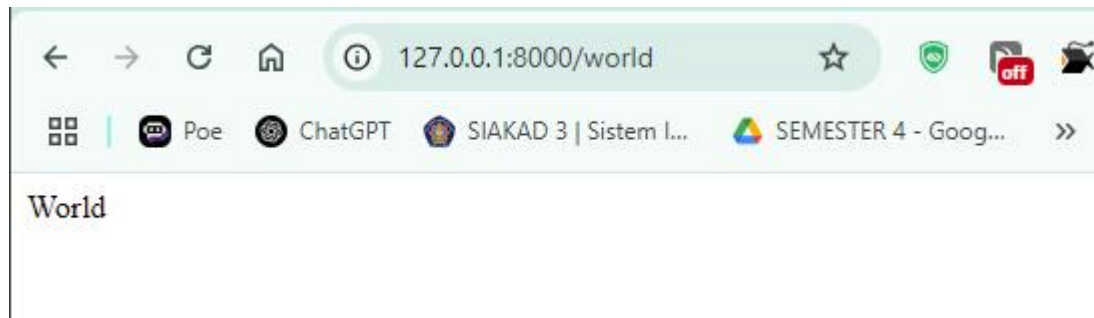
```
use Illuminate\Support\Facades\Route;

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

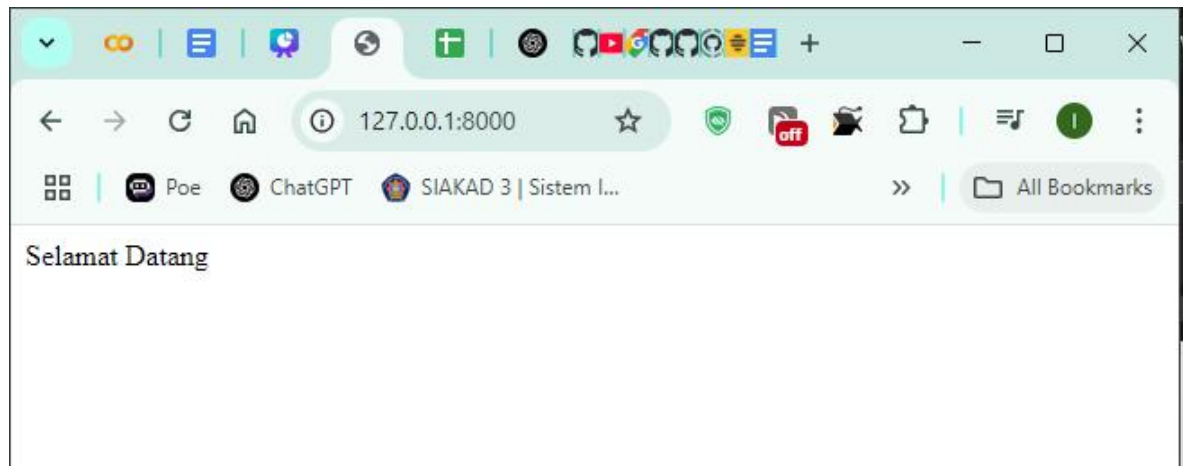
- e. Bukalah pada browser, tuliskan URL untuk memanggil route tersebut:



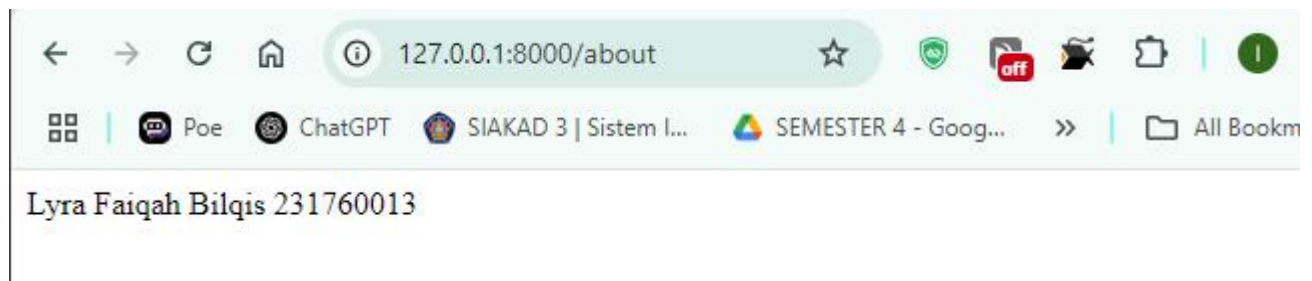
localhost/PWL_2024/public/world. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda. Kode di atas adalah konfigurasi routing dalam Laravel yang menentukan bagaimana aplikasi merespons permintaan URL tertentu. Route "/" menampilkan halaman utama dengan tampilan 'welcome'. Route "/hello" menampilkan teks "Hello World", sedangkan route "/world" menampilkan teks "World". Dengan menambahkan route '/world', pengguna dapat mengakses '127.0.0.1:8000/world' untuk melihat teks "World" langsung di browser.



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.



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>

Langkah-langkah Praktikum:

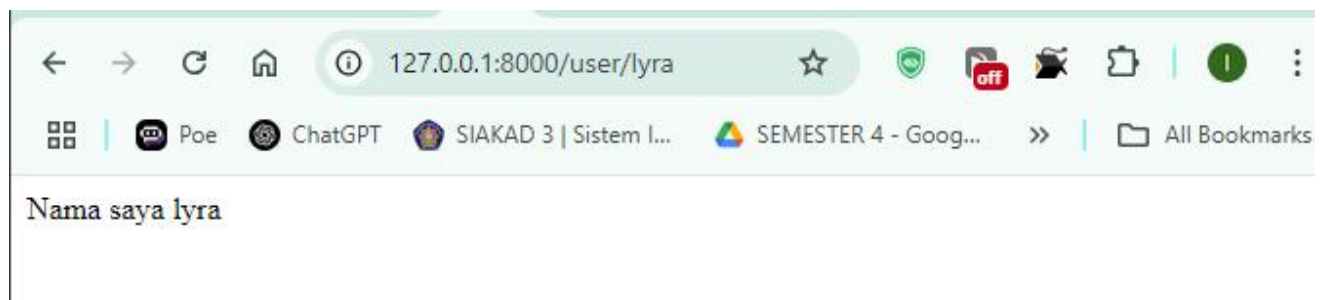
Untuk membuat routing dengan parameter dapat dilakukan dengan cara berikut ini.



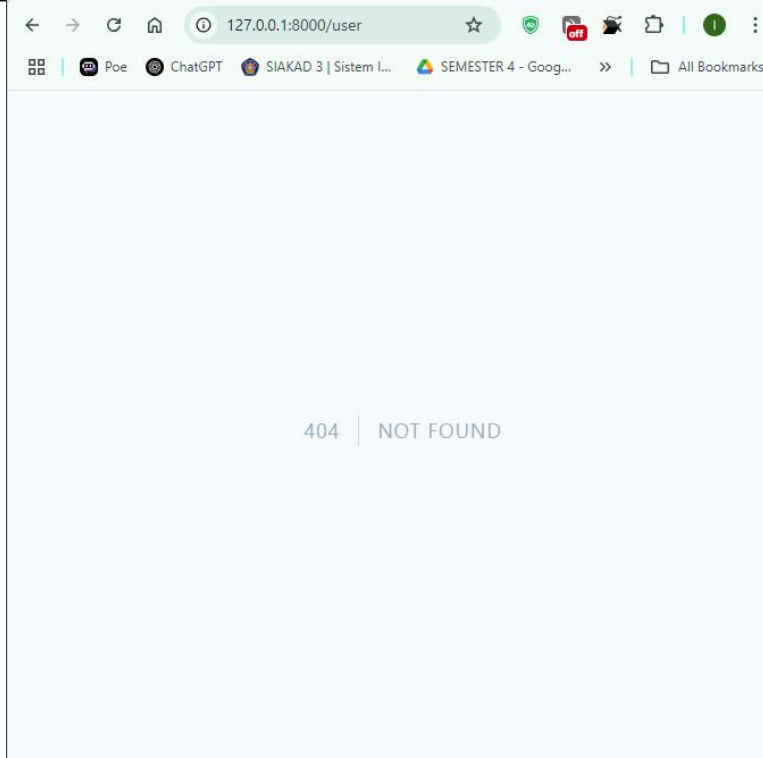
- a. Kita akan memanggil route `/user/{name}` sekaligus mengirimkan parameter berupa nama user `$name` seperti kode di bawah ini.

```
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. Route `"/user/{name}"` akan menampilkan teks "Nama saya [name]" sesuai dengan nilai yang diberikan pada parameter `{name}`. Misalnya, jika pengguna mengakses `/user/Lyra`, maka browser akan menampilkan "Nama saya Lyra". Parameter `{name}` bersifat dinamis, sehingga dapat berubah sesuai input yang diberikan di URL.



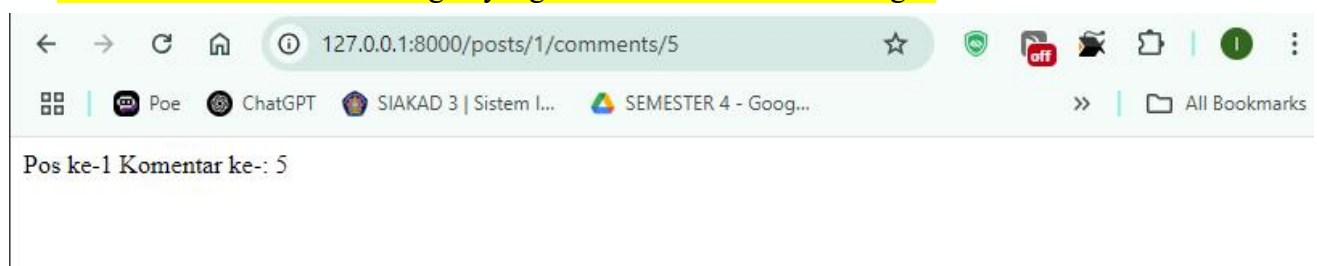
- c. Selanjutnya, coba tuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda. Jika hanya mengakses `/user/` tanpa memberikan nilai `{name}`, Laravel akan menampilkan error 404 (Not Found) karena route tersebut mengharuskan adanya parameter `name`. Tanpa nilai yang sesuai, Laravel tidak dapat memproses permintaan, sehingga halaman tidak ditemukan.



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

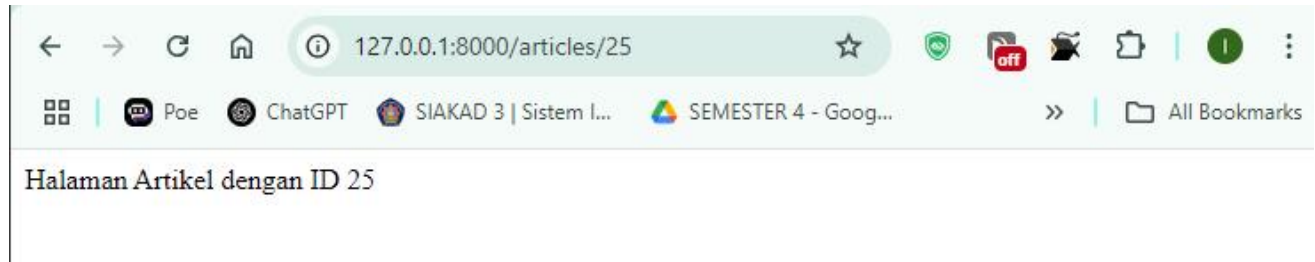
```
Route::get('/posts/{post}/comments/{comment}', function ($postId, $commentId) {  
    return 'Pos ke-' . $postId . " Komentar ke-: " . $commentId;  
});
```

- 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. Jika route `/posts/{post}/comments/{comment}` dijalankan dengan parameter, misalnya `/posts/1/comments/5`, maka output yang muncul adalah "Pos ke-1 Komentar ke-: 5". Laravel akan mengambil nilai `{post}` sebagai `$postId` dan `{comment}` sebagai `$commentId`, lalu menampilkannya dalam format teks sesuai dengan yang didefinisikan di dalam fungsi.





- 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.

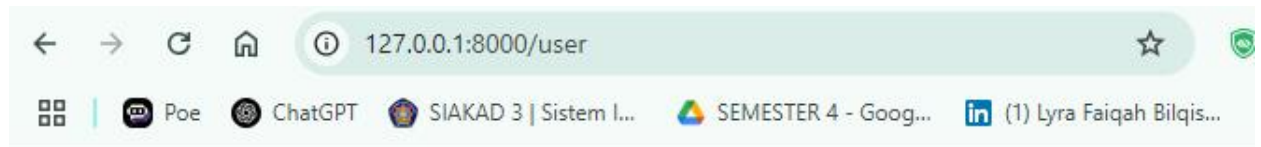
- a. Kita akan memanggil route `/user` sekaligus mengirimkan parameter berupa nama user `$name` dimana parameternya bersifat opsional.

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



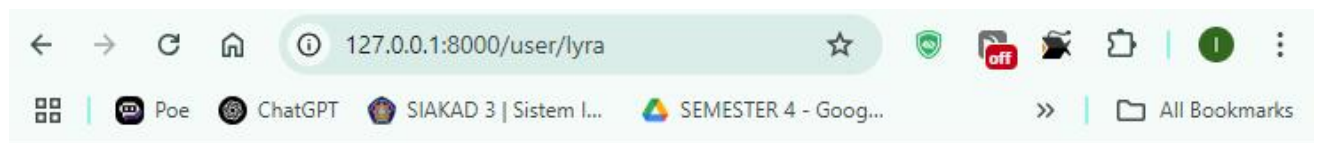
```
return 'Nama saya '.$name;
});
```

- b. Jalankan kode dengan menuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda. Variabel \$name tidak diberikan nilai default saat nilainya null. Dalam PHP, jika null dikonversi ke string, hasilnya adalah string kosong (""), sehingga output yang ditampilkan hanya "Nama saya" tanpa tambahan teks lainnya.



Nama saya

- c. Selanjutnya tuliskan URL: **localhost/PWL_2024/public/user>NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda. Penambahan route **/user/{name?}** memungkinkan parameter name menjadi opsional. Jika pengguna mengakses **/user/{name}**, maka sistem akan menampilkan "Nama saya {name}". Namun, jika parameter tidak diberikan, misalnya hanya mengakses **/user/**, maka sistem akan menampilkan "Nama saya tidak diketahui".



Nama saya lyra

- 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. Mengakses **/user/** tanpa memberikan parameter, maka yang muncul adalah "Nama saya John". Hal ini terjadi karena dalam deklarasi fungsi, \$name sudah diberikan nilai default 'John', sehingga jika parameter tidak diberikan, sistem otomatis menggunakan nilai default tersebut.



- **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
```

- 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;

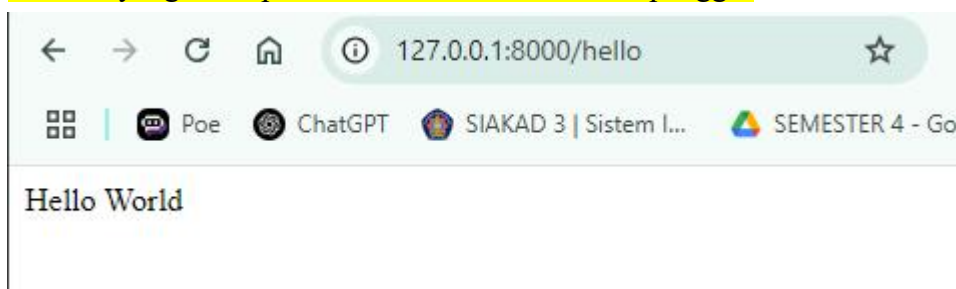
use Illuminate\Http\Request;

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. Kode `'Route::get('/hello', [WelcomeController::class, 'hello']);'` menghubungkan URL `'/hello'` ke metode `'hello()'` di `'WelcomeController'`. Saat diakses, metode ini mengembalikan "Hello World". Di `'WelcomeController.php'`, namespace memastikan lokasi controller, sedangkan metode `'hello()'` menampilkan teks "Hello World", yang ditampilkan di browser saat route dipanggil.



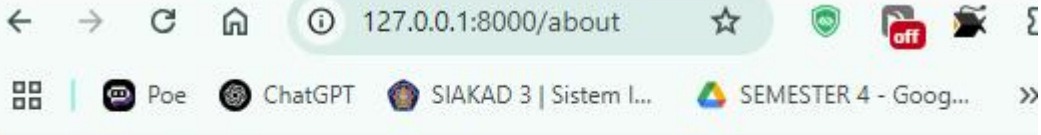


- f. Modifikasi hasil pada praktikum poin 2 (Routing) dengan konsep controller. Pindahkan logika eksekusi ke dalam controller dengan nama PageController.

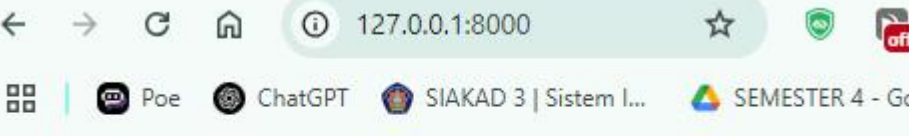
Resource	POST	GET	PUT	DELETE
/		Tampilkan Pesan 'Selamat Datang' PageController : index		
/about		Tampilkan Nama dan NIM PageController : about		
/articles/ {id}		Tampilkan halaman dinamis 'Halaman Artikel dengan Id {id}' id diganti sesuai input dari url		



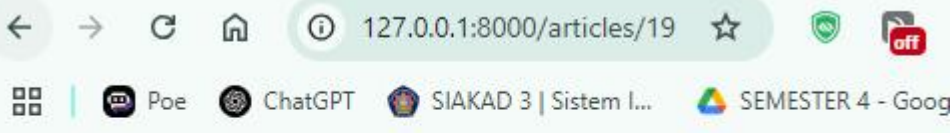
PageController : articles



Nama dan NIM

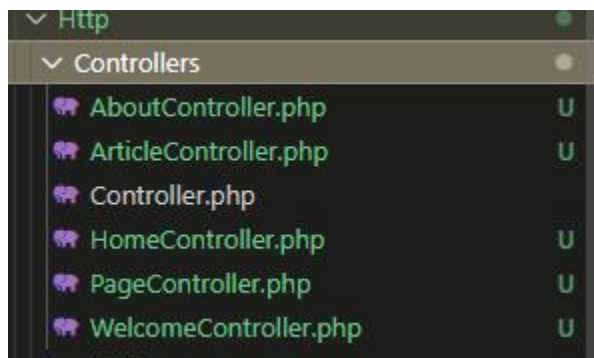


Selamat Datang



PageController : articles - Halaman Artikel dengan ID 19

- 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.



<?php



```
use Illuminate\Support\Facades\Route;  
use App\Http\Controllers\WelcomeController;  
use App\Http\Controllers\PageController;  
use App\Http\Controllers\HomeController;  
use App\Http\Controllers\AboutController;  
use App\Http\Controllers\ArticleController;
```

```
/*  
|-----  
| Web Routes  
|-----  
|  
| Here is where you can register web routes for your application. These  
| routes are loaded by the RouteServiceProvider and all of them will  
| be assigned to the "web" middleware group. Make something great!  
|  
*/  
// Route untuk halaman utama  
Route::get('/', function () {  
    return 'Selamat Datang';  
});
```

```
// Route untuk menampilkan "Hello World"  
Route::get('/hello', [WelcomeController::class, 'hello']);
```

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

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

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

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

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

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



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

```
Route::get('/', HomeController::class);  
Route::get('/about', AboutController::class);  
Route::get('/articles/{id}', ArticleController::class);
```

- 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.

- 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/create | photos.store | App\Http\Controllers\PhotoController@store | web |
| | GET HEAD | photos/{photo} | 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 |



```
λ Cmler
λ php artisan route:list

GET|HEAD / ..... HomeController
POST _ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > Execut
GET|HEAD _ignition/health-check ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckCon
POST _ignition/update-config ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfig
GET|HEAD about ..... AboutController
GET|HEAD api/user .....
GET|HEAD articles/{id} ..... ArticleController
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 > CsrfCookieController@show
GET|HEAD user/{name?} .....
GET|HEAD user/{name} .....
GET|HEAD world .....

Showing [20] 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`.

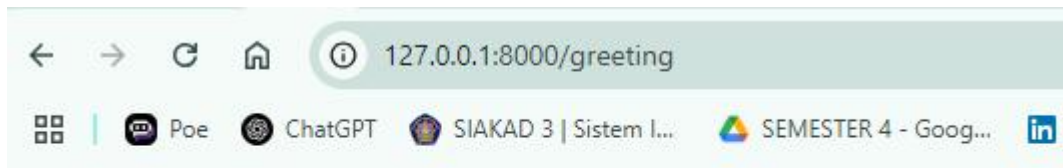
```
<!-- View pada resources/views/hello.blade.php -->
<html>
    <body>
        <h1>Hello, {{ $name }}</h1>
    </body>
</html>
```

- 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. **Laravel akan memproses routing dan menampilkan tampilan dari file `hello.blade.php` yang ada di direktori `resources/views`. Dalam file tersebut, variabel `$name` digunakan untuk menampilkan nama yang dikirim dari route. Karena route mengirimkan nilai 'Lyra' untuk variabel `name`, output yang ditampilkan di halaman akan berupa teks "Hello, Lyra" dalam format heading (`<h1>`).**



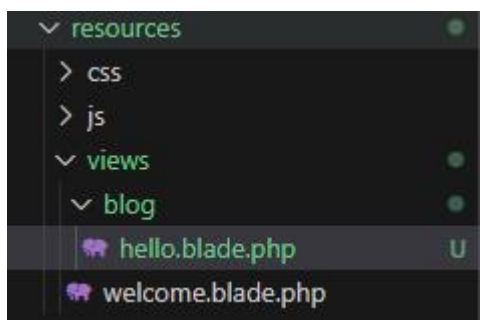
Hello, Lyra

- 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']);  
});
```

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



sesuai gambar, Laravel akan mencari file tampilan `hello.blade.php` di dalam folder `'blog'`, yang berada di dalam direktori `'resources/views'`. Saat pengguna mengakses URL `'/greeting'`, Laravel akan menampilkan tampilan dari file `'blog.hello'`, yang berisi teks "Hello, Andi" dalam elemen `'<h1>'`. Ini terjadi karena route telah diperbarui untuk merujuk ke tampilan dalam subdirektori `'blog'`, dan variabel `'$name'` yang dikirim masih berisi nilai `'Andi'`.



Hello, Andi

- 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:

- Buka `WelcomeController.php` dan tambahkan fungsi baru yaitu `greeting`.

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

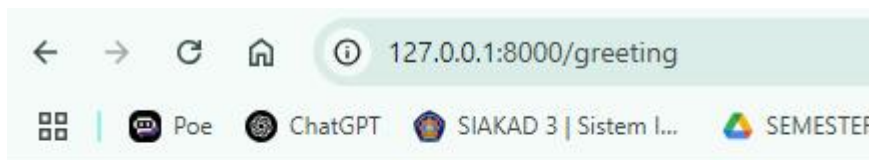
    public function greeting(){
        return view('blog.hello', ['name' => 'Andi']);
    }
}
```




- 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. **ketika pengguna mengakses URL /greeting, Laravel akan memanggil fungsi greeting() dalam WelcomeController. Fungsi ini akan merender tampilan blog.hello dari dalam direktori resources/views/blog, dan mengirimkan variabel \$name dengan nilai 'Andi'. Akibatnya, halaman yang ditampilkan akan berisi teks "Hello, Andi" dalam elemen <h1>.**



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. Setelah modifikasi seperti pada gambar, ketika pengguna mengakses URL '/greeting', Laravel akan menjalankan fungsi 'greeting()' dalam 'WelcomeController'. Fungsi ini akan merender tampilan 'blog.hello' dan mengirimkan dua variabel: '\$name' dengan nilai 'Andi' dan '\$occupation' dengan nilai 'Astronaut'. Teks tersebut ditampilkan dalam elemen '<h1>', sehingga muncul dalam ukuran besar di halaman. Jika terjadi kesalahan seperti variabel tidak dikenali, maka Laravel bisa menampilkan error terkait.



Hello, Andi

You are Astronaut

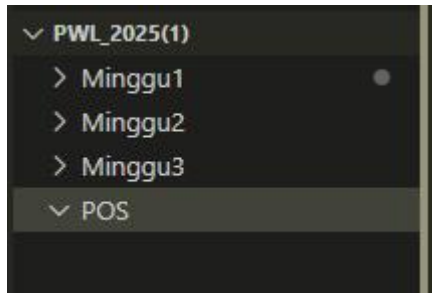
Simpan perubahan yang telah dilakukan pada Git.

SOAL PRAKTIKUM

1. Jalankan Langkah-langkah Praktikum pada jobsheet di atas. Lakukan sinkronisasi perubahan pada project PWL_2024 ke Github. **Sudah.**



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



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. **Sudah**



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>

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