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

Nama : Adinda Ivanka Maysanda Putri

Kelas : SIB 2B

NIM : 2341760058

JOBSHEET 02

ROUTING, CONTROLLER, DAN VIEW

- **Routing**

1. Basic Routing

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

Langkah-langkah Praktikum:

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

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

Kita akan menggunakan project minggu sebelumnya yaitu PWL_2024.

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

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

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

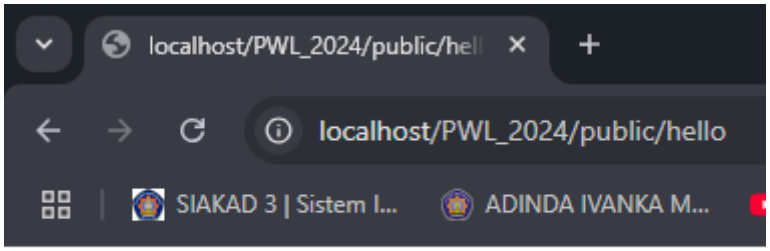
_

```
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.

- 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



Hello World

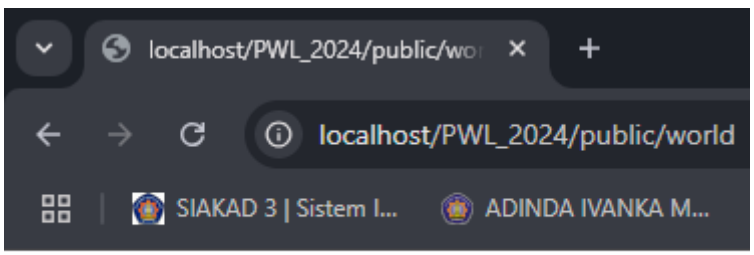
d. Untuk membuat route kedua, tambahkan route /world seperti di bawah ini:

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

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

-
```
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.

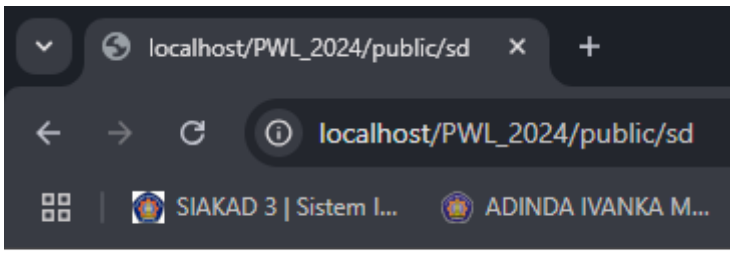
- fungsi get akan mengarahkan kita ke Output World, ketika mengakses domain / world, sesuai dengan nilai return yang sudah diisikan.



World

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

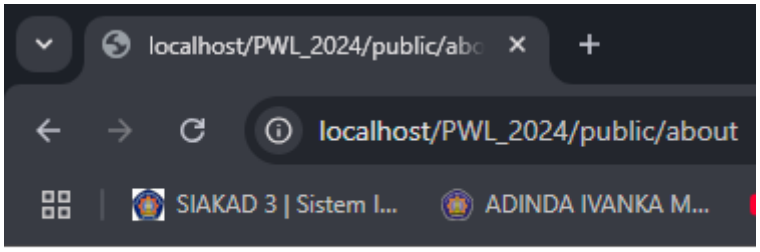
```
Route::get('/sd', function() {
    return 'Selamat Datang';
});
```



Selamat Datang

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

```
Route::get('/about', function() {
    return 'Nama : Adinda Ivanka Maysanda Putri,
    NIM : 2341760058';
});
```



Nama : Adinda Ivanka Maysanda Putri, NIM : 2341760058

2. Route Parameters

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

Langkah-langkah Praktikum:

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

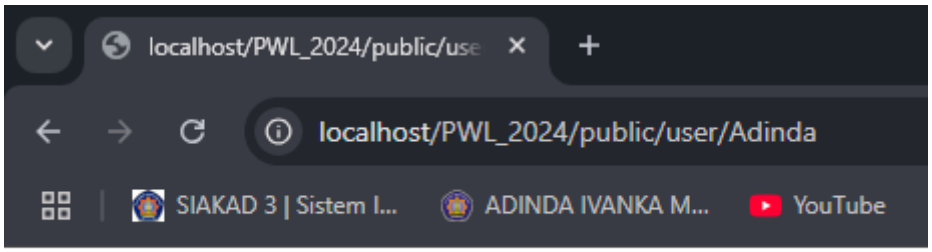
a. Kita akan memanggil route */user/{name}* sekaligus mengirimkan parameter berupa nama user $name seperti kode di bawah ini.

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

-
```php
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.

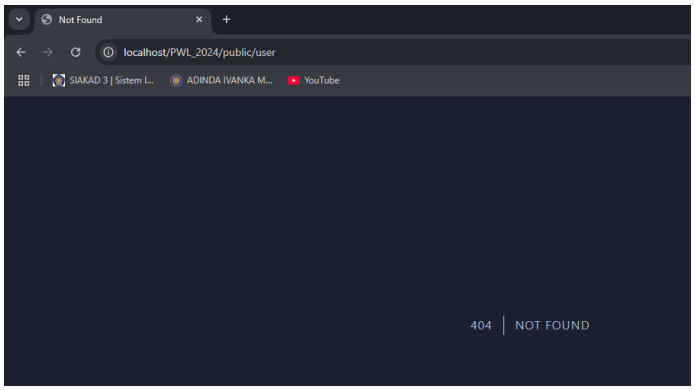
- Route tersebut Bisa menampung value yang ditulis user karena memiliki parameter {name}, dan akan dikembalikan melalui return berupa String



Nama saya Adinda

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

- Dikarenakan parameter {name} null, output yang dihasilkan website tidak bisa ditemukan.



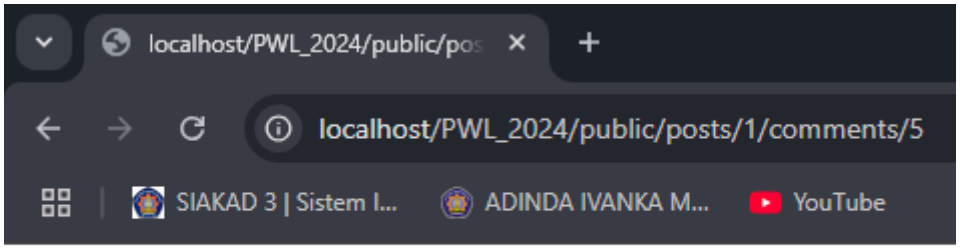
d. Suatu route, juga bisa menerima lebih dari 1 parameter seperti kode berikut ini. Route menerima parameter $postId dan juga $comment.

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

-
```php
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.

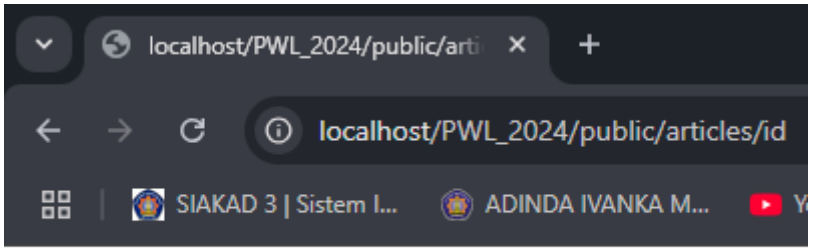
- kita bisa menginput lebih dari satu data pada domain tersebut, Seperti pada hasil yang terlihat, kita bisa melakukan input pada posts, dan comments dan hasil yang ditampilkan sesuai dengan input kita pada domain tersebut.



Pos ke-1 Komentar ke-: 5

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

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



Halaman Artikel dengan ID : id

-

3. 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.

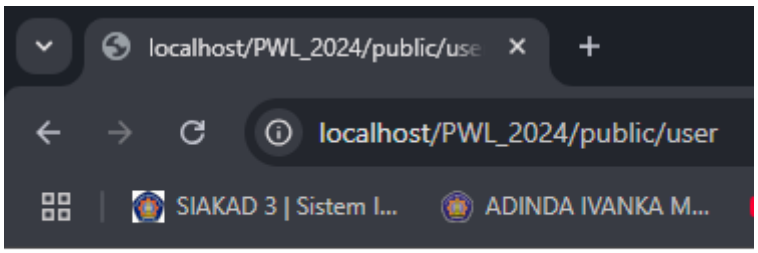
a. Kita akan memanggil route */user* sekaligus mengirimkan parameter berupa nama user

$name dimana parameternya bersifat opsional.

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

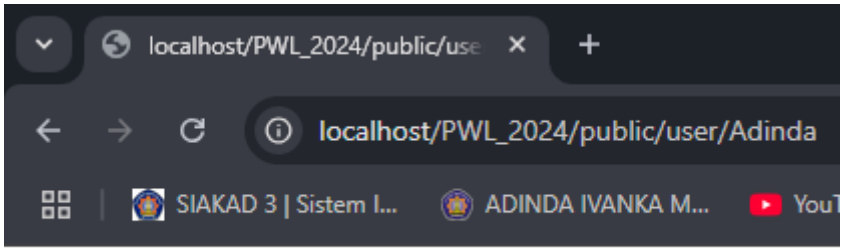
```
Route::get('/user/{name?}', function ($name=null) {
    return 'Nama saya '.$name;
});
```

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



Nama saya

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



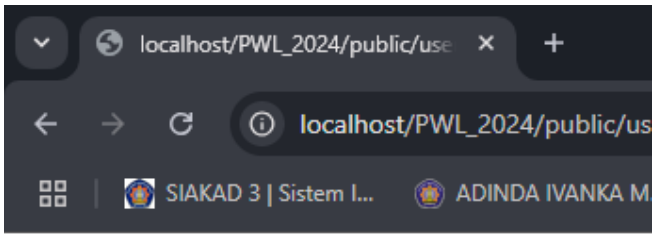
Nama saya Adinda

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

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

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

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



Nama saya John

-

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.

Simpan perubahan yang telah dilakukan pada Git.

- **Controller**

- **Membuat Controller**

Langkah-langkah Praktikum:

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

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

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

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

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
{
    //
}
```

-

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

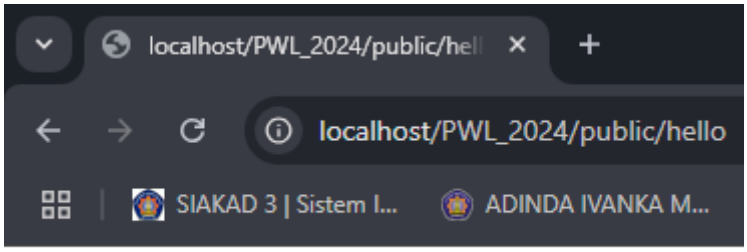
-
```
public function hello() {
    return 'Hello World';
}
```

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']);
```

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

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



-
Hello World

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

Resource	POST	GET	PUT	DELETE
/		Tampilkan Pesan ‘Selamat Datang’ PageController : index		
/about		Tampilkan Nama dan NIM PageController : about		

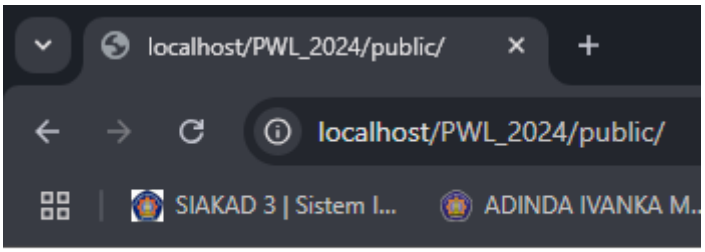
/articles/ {id}		Tampilkan halaman dinamis ‘Halaman Artikel dengan Id {id}’ id diganti sesuai input dari url		
		PageController : articles		

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

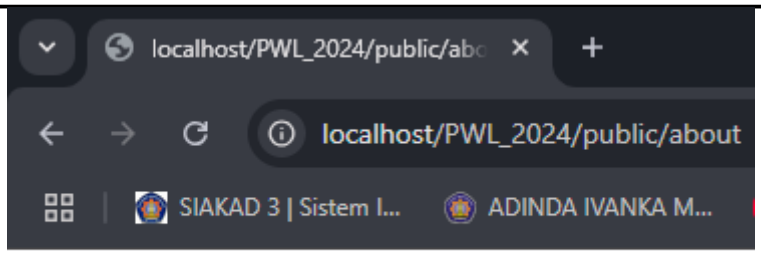
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

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

    public function about() {
        return 'Nama : Adinda Ivanka Maysanda Putri || NIM : 2341760058';
    }

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

routes > web.php > ...

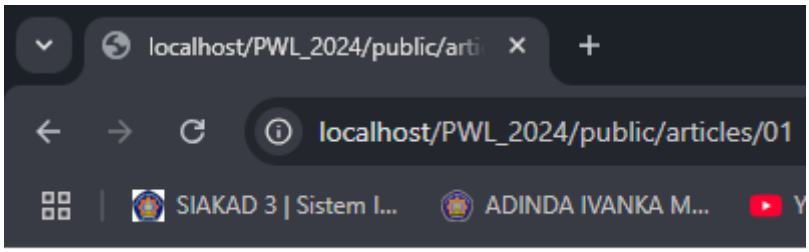
```php
Route::get('/', [PageController::class, 'index']);
Route::get('/about', [PageController::class, 'about']);
Route::get('/articles/{id}', [PageController::class, 'articles']);
```



Selamat Datang



Nama : Adinda Ivanka Maysanda Putri || NIM : 2341760058



Halaman artikel dengan ID : 01

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:\laragon\www\PWL_2024\app\Http\Controllers\HomeController.php] created successfully.

PS C:\laragon\www\PWL_2024> php artisan make:controller AboutController

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

PS C:\laragon\www\PWL_2024> php artisan make:controller ArticleController

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

app > Http > Controllers > HomeController.php > ...

```php
class HomeController extends Controller
{
    public function index() {
        return 'Selamat Datang';
    }
}
```

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

```php

class AboutController extends Controller
{
    public function about() {
        return 'Nama : Adinda Ivanka Maysanda Putri || NIM : 2341760058';
    }
}
```

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

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

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

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

- **Resource Controller**

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

Langkah-langkah Praktikum:

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

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

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

-
```
PS C:\laragon\www\PWL_2024> php artisan make:controller PhotoController --resource

  INFO  Controller [C:\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);
```

-
```
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

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

  GET|HEAD        / ..... HomeController@index
  POST            _ignition/execute-solution ignition.executeSolution › Spatie\Lar…
  GET|HEAD        _ignition/health-check ignition.healthCheck › Spatie\LaravelIgni…
  POST            _ignition/update-config ignition.updateConfig › Spatie\LaravelIg…
  GET|HEAD        about ..... AboutController@about
  GET|HEAD        api/user .....
  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 › Csrf…
  GET|HEAD        sd .....
  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'
]);
```

-

```
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.

- **View**

- **Membuat View**

Langkah-langkah Praktikum:

1. 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>
```

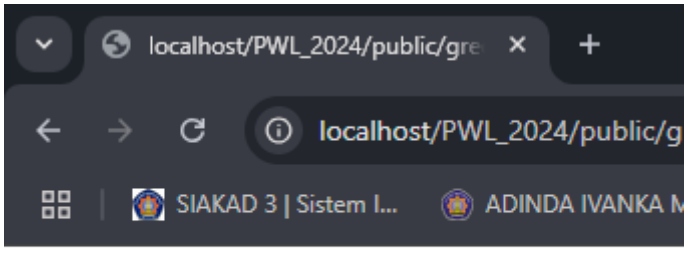
-

2. 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'=>'Adinda']);
});
```

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



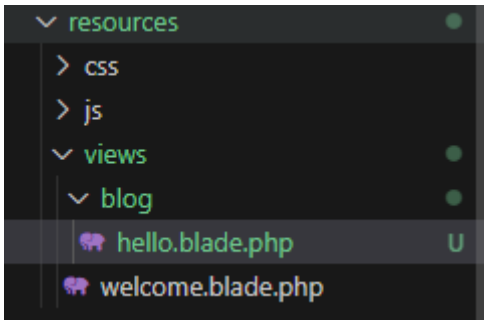
Hello, Adinda

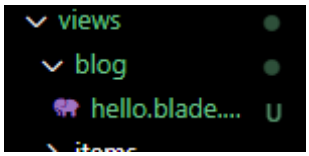
- **View dalam direktori**

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

Langkah-langkah Praktikum:

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





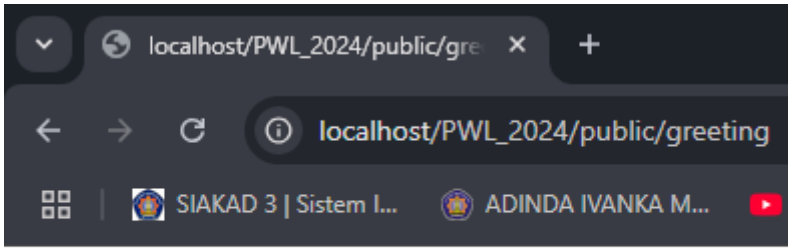
c. Selanjutnya lakukan perubahan pada route.

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

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

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

- Dari hasil pengamatan tetap bisa diakses walaupun pada url seharusnya tidak bisa diakses. Hal ini terjadi dikarenakan pada route kita telah mengidentifikasi path dari file tersebut. Jika, terdapat sebuah direktori didalam view. Maka, pada route kita identifikasi direktori tersebut lalu nama filenya agar bisa diakses tanpa harus mencantumkan nama direktorinya.



Hello, Adinda

- **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']);
    }
}
```

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

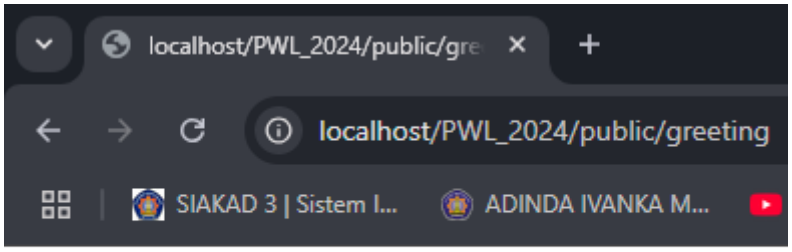
b. Ubah route /greeting dan arahkan ke WelcomeController pada fungsi greeting.

```
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.

- Dari hasil pengamatan diatas. Kita dapat mengetahui perbedaan pemanggilan file tersebut. Pada praktikum sebelumnya tanpa menggunakan controller. Pada praktikum kali ini menggunakan controller dengan cara membuat sebuah fungsi terlebih dahulu pada controller yang nantinya akan me-return atau memanggil file hello.blade.php pada direktori blog. Lalu, controller tersebut dipanggil dengan menggunakan route dengan cara ketika menjalankan url /greeting . maka, akan memanggil file controller yaitu WelcomeController



Hello, Adinda

- **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');
        }
}
```

-
```
    public function gretting() {
        return view('blog.hello')
        -> with('name', 'Adinda')
        -> 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>
```

-
resources > views > blog > hello.blade.php > ...
```
1   <html>
2   <body>
3   <h1>Hello, {{ $name }}</h1>
4   <h1>You are {{ $occupation }}</h1>
5   </body>
6   </html>
```

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

- Dari hasil pengamatan berikut kita dapat mengetahui untuk meneruskan data ke dalam view. Selain menggunakan array pada praktikum sebelumnya. Terdapat alternatif lain yaitu menggunakan with. Dengan mencantumkan with kita bisa melakukan input data ke dalam parameter name dan occupation pada kasus diatas. Yang nantinya akan masuk data tersebut ke dalam view dan ditampilkan pada halaman website. Seperti pada gambar berikut.

Simpan perubahan yang telah dilakukan pada Git.

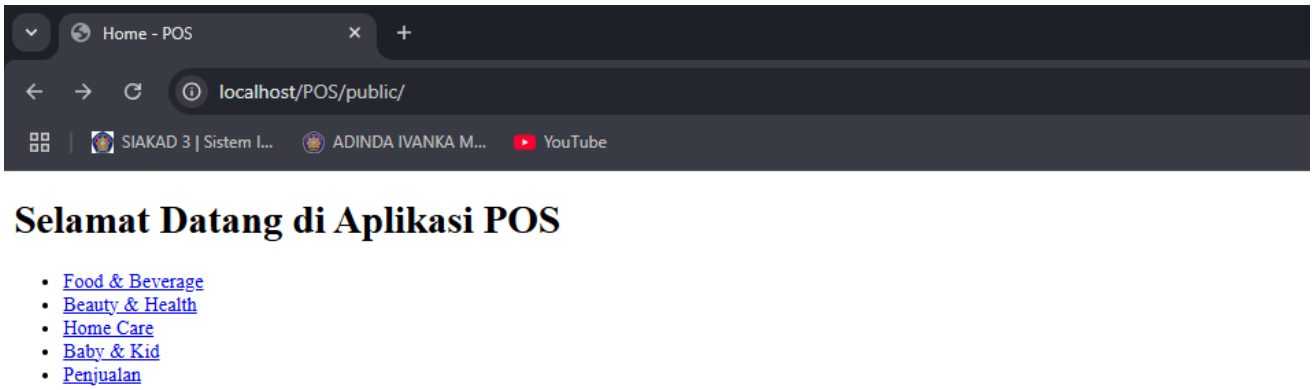
SOAL PRAKTIKUM

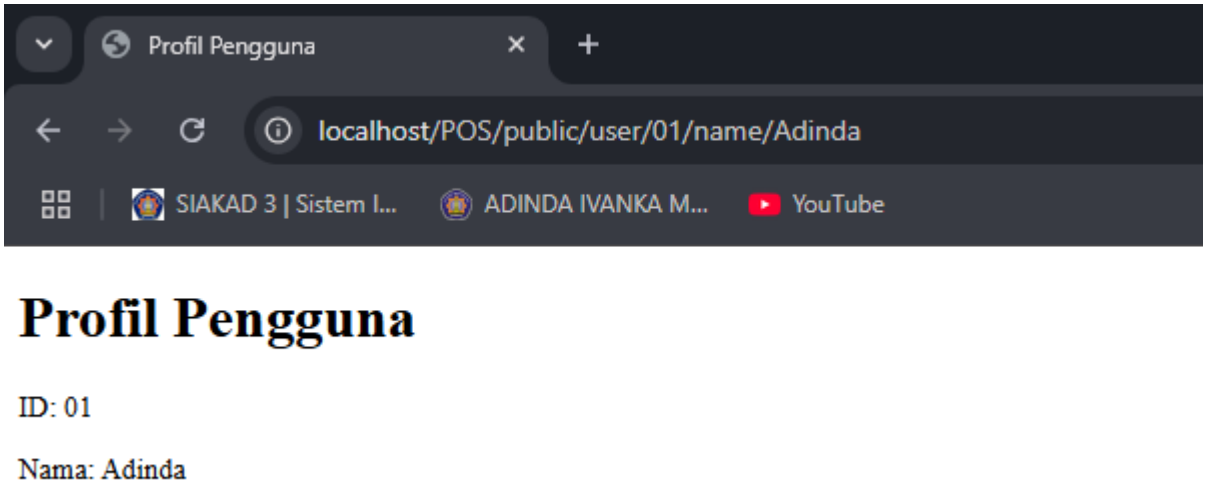
1. Jalankan Langkah-langkah Praktikum pada jobsheet di atas. Lakukan sinkronisasi perubahan pada project PWL_2024 ke Github.
2. Buatlah project baru dengan nama POS. Project ini merupakan sebuah aplikasi Point of Sales yang digunakan untuk membantu penjualan.
   ⇨ https://github.com/adindaivankamp/POS
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.





Link PWL_2024 = https://github.com/adindaivankamp/PWL_2024

Link POS = https://github.com/adindaivankamp/POS