PRAKTIKUM

PEMROGRAMAN FRAMEWORK

MODUL 3

Routing dan Bundling Asset di Laravel

Sistem Informasi
Telkom University
Surabaya

Disusun Oleh:

Meike Syahra Yundhi Afie

1204220093/IS-05-02

PROGRAM STUDI S1 SISTEM INFORMASI

FAKULTAS REKAYASA INDUSTRI

TELKOM UNIVERSITY SURABAYA

2024

Praktik Laravel Routing

Buka file **routes/web.php,** buat sebuah Route dengan menuliskan kode program seperti di bawah ini.

```
1. Route::get('/routing', function() {
2.     return view('routing');
3. });
```

Penjelasan:

Code di atas sebuah route menggunakan method get(). Route akan merespons permintaan get ke URL'/routing' yang diakses menggunakan fungsi anonym yang akan dijalankan sehingga mengembalikan sebuah view dengan nama routing.

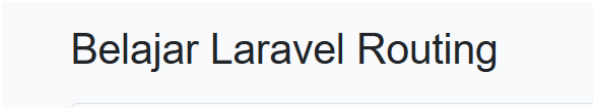
Buat file View baru pada direktori **/resources/views/** dengan nama **routing.blade.php**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Belajar Laravel Routing</title>
    @vite('resources/sass/app.scss')
</head>
<body>
    <div class="container m-5">
        <h1>Belajar Laravel Routing</h1>

        <div class="list-group list-group-numbered mt-4">
            {{-- Kode anda selanjutnya letakkan di sini --}}

        </div>

        {{-- Khusus kode program untuk Route Groups di sini --}}
    </div>
    @vite('resources/js/app.js')
</body>
</html>
```

Penjelasan:

Pada file view baru diisi sebuah code HTML yang memiliki class container m-5 berisi judul halaman dan sebuah div dengan class list-group list-group-numbered mt-4 yang akan diisi dengan data yang diperoleh dari route. Tag @vite() untuk mengkomplikasi file css dan file JavaScript.

Output:



Praktik Basic Routing

Buka file **routes/web.php,** praktikkan **Basic Routing (No View, No Controller)** dengan menuliskan kode program seperti di bawah ini.

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

Penjelasan:

Kode tersebut adalah bagian dari proyek Laravel yang mendefinisikan route untuk URL '/basic_routing'. Saat URL tersebut diakses dengan metode HTTP GET, server akan merespons dengan string 'Hello World'. Ini adalah cara yang cepat dan sederhana untuk memberikan respons dasar dari server saat URL tertentu diakses.

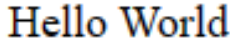
Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/basic_routing') }}" class="list-group-item list-group-item-
action">
                Basic Routing (No View, No Controller)
            </a>
```

Penjelasan:

Kode tersebut adalah bagian dari tampilan Blade dalam proyek Laravel yang membuat hyperlink menuju URL '/basic_routing'. Kelas CSS diterapkan untuk memberikan gaya tampilan, sementara teks di dalam hyperlink menjelaskan bahwa ini adalah rute dasar tanpa tampilan atau kontroler. Ini adalah cara yang efektif untuk membuat navigasi dalam aplikasi Laravel.

Output:



Praktik View Route

Buka file **routes/web.php,** praktikkan **View Route** dengan menuliskan kode program seperti di bawah ini.

```
Route::view('/view_route', 'view_route');
Route::view('/view_route', 'view_route', ['name' => 'Purnama']);
```

Penjelasan:

Kode tersebut menggunakan method view() dari class Route dalam proyek Laravel untuk menetapkan route ke URL '/view_route'. Saat URL tersebut diakses, Laravel akan menampilkan view 'view_route'. Pada baris kedua, tambahan array ['name' => 'Purnama'] mengirimkan data 'Purnama' ke view tersebut. Ini memungkinkan tampilan untuk menampilkan informasi perlu menuliskan logika controller terpisah yang memudahkan dalam menangani tampilan sederhana di Laravel.

Buat file View dengan nama **view_route.blade.php**, kemudian isikan dengan kode program seperti di bawah ini.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>View Route</title>
    @vite('resources/sass/app.scss')
</head>
<body>
    <div class="container m-5">
        <h1>This is from View Route</h1>
        <p>Hello, My name is {{ $name }}</p>
    </div>
    @vite('resources/js/app.js')
</body>
</html>
```

Penjelasan:

Kode HTML tersebut adalah sebuah tampilan Blade dalam proyek Laravel. Ini menetapkan metadata dasar, termasuk charset, viewport, dan judul halaman. File ini menggunakan directive Blade @vite() untuk menghubungkan file SASS untuk kompilasi sebelum rendering. Di dalam elemen <body>, terdapat tata letak dasar halaman dengan judul dan sebuah paragraf yang menampilkan nilai dari variabel $name. Di akhir file, directive Blade @vite() digunakan lagi untuk menghubungkan file JavaScript untuk kompilasi. Ini adalah cara yang efisien untuk mengelola tampilan dan aset dalam proyek Laravel.

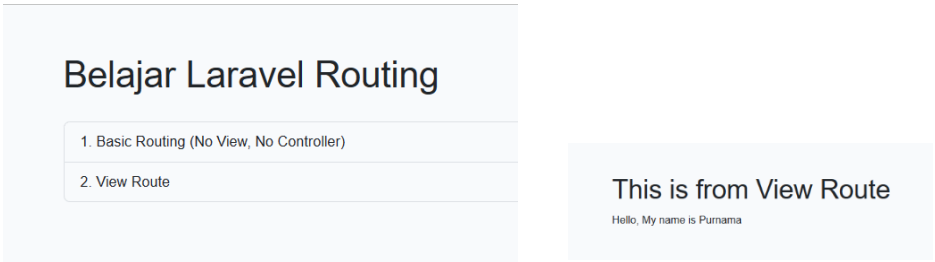
Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/view_route') }}" class="list-group-item list-group-item-
action">
                View Route
            </a>
```

Penjelasan:

Kode tersebut adalah bagian dari sebuah tampilan Blade dalam proyek Laravel. Ini adalah sebuah hyperlink yang menggunakan directive Blade {{ url('/view_route') }} untuk menentukan URL tujuan, yaitu '/view_route'. Ketika hyperlink ini diklik, pengguna akan diarahkan ke URL tersebut. Kelas "list-group-item list-group-item-action" diterapkan untuk memberikan gaya tampilan dari Bootstrap, yang membuat hyperlink ini terlihat seperti sebuah elemen dalam sebuah daftar dengan aksi. Teks di dalam hyperlink adalah "View Route", yang menjelaskan tujuan dari hyperlink tersebut. Ini adalah cara yang efektif untuk membuat navigasi antarhalaman dalam aplikasi web Laravel.

Output:



Praktik Controller Route

Generate file controller dengan nama RouteController menggunakan bantuan script artisan sebagai berikut: **php artisan make:controller RouteController**

```
PS C:\xampp\htdocs\basic-router> php artisan make:controller RouteController

  INFO  Controller [C:\xampp\htdocs\basic-router\app\Http\Controllers\RouteController.php] created successfully.

PS C:\xampp\htdocs\basic-router>
```

Penjelasan:

Perintah yang diberikan adalah php artisan make:controller RouteController. Ketika perintah ini dijalankan dalam terminal proyek Laravel, itu akan membuat file controller baru dengan nama "RouteController". File ini akan ditempatkan dalam direktori "app/Http/Controllers". Dengan memiliki controller yang dihasilkan, Anda dapat menulis logika untuk menangani permintaan HTTP dari rute-rute yang telah Anda tentukan dalam proyek Laravel Anda.

Tambahkan function bernama **“index”** pada class **RouteController** tersebut.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class RouteController extends Controller
{
    public function index() {
        return "This is from Controller";
    }
}
```

Penjelasan:

Function ini mengembalikan sebuah string "This is from Controller" ketika dipanggil. Class "RouteController" terletak dalam namespace "App\Http\Controllers" dan meng-extend class "Controller" bawaan Laravel. Dengan menambahkan function "index" ke dalam controller, Anda dapat menangani permintaan HTTP yang datang ke rute yang sesuai dengan function ini. Saat rute tersebut diakses, Laravel akan menjalankan function "index" dari "RouteController" dan mengembalikan respons yang diinginkan, dalam hal ini sebuah string.

Buka file **routes/web.php,** praktikkan **Controller Route** dengan menuliskan kode program seperti di bawah ini.

```php
Route::get('/controller_route', [RouteController::class, 'index']);
```

Penjelasan:

Baris kode tersebut menetapkan sebuah route dalam proyek Laravel untuk URL '/controller_route'. Saat URL tersebut diakses dengan metode HTTP GET, Laravel akan mengarahkan permintaan tersebut ke function 'index' dalam controller 'RouteController'. Ini adalah cara yang umum digunakan untuk menghubungkan route dengan function dalam controller dalam proyek Laravel.

Pada file **routes/web.php**, lihat bagian atas file tersebut, pastikan **RouteController** ter-import seperti kode program di bawah ini.

```php
<?php

use App\Http\Controllers\RouteController;
use Illuminate\Support\Facades\Route;
```

Penjelasan:

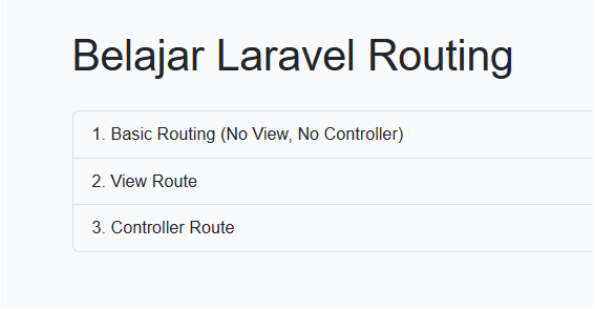
Kode di atas mengimpor class RouteController dari namespace App\Http\Controllers dan class Route dari namespace Illuminate\Support\Facades. Ini memungkinkan penggunaan class tersebut tanpa menyertakan namespace lengkap setiap kali digunakan dalam file atau skrip PHP. Ini adalah cara yang umum digunakan dalam pengembangan aplikasi Laravel untuk mengorganisir dan mengelola class yang digunakan.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/controller_route') }}" class="list-group-item list-group-item-action">
                Controller Route
            </a>
```

Penjelasan:
Kode di atas merupakan bagian dari sebuah hyperlink dalam file tampilan Blade pada proyek Laravel. Ketika hyperlink ini diklik, pengguna akan diarahkan ke URL '/controller_route'. Kelas CSS "list-group-item list-group-item-action" diterapkan untuk memberikan gaya tampilan sesuai dengan framework CSS Bootstrap, sehingga hyperlink ini terlihat seperti elemen dalam daftar dengan aksi. Teks di dalam hyperlink adalah "Controller Route", yang memberikan deskripsi tentang tujuan dari hyperlink tersebut. Ini adalah cara yang umum digunakan untuk membuat tautan antarhalaman dalam aplikasi web Laravel.

Output:



Praktik Rediect Route

Buka file **routes/web.php,** praktikkan **Redirect Route** dengan menuliskan kode program seperti di bawah ini.

```
Route::redirect('/', '/routing');
```

Penjelasan:

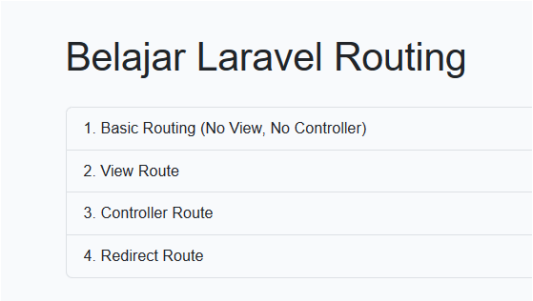
Kode di atas adalah bagian dari file routes/web.php dalam proyek Laravel. Ini menetapkan suatu redirect untuk URL root ('/') agar mengarahkan pengguna langsung ke URL '/routing'. Dengan demikian, saat pengguna mengakses URL root, mereka akan secara otomatis dialihkan ke halaman yang ditetapkan dalam redirect ini, yaitu '/routing'. Ini adalah cara yang umum digunakan untuk mengarahkan pengguna ke halaman awal atau halaman utama dalam sebuah aplikasi web Laravel.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/') }}" class="list-group-item list-group-item-action">
                Redirect Route
            </a>
```

Penjelasan:
Kode tersebut merupakan bagian dari sebuah hyperlink dalam file tampilan Blade pada proyek Laravel. Saat hyperlink ini diklik, pengguna akan diarahkan kembali ke URL root ('/'). Kelas CSS "list-group-item list-group-item-action" diterapkan untuk memberikan gaya tampilan dari Bootstrap, sehingga hyperlink ini terlihat seperti elemen dalam daftar dengan aksi. Teks di dalam hyperlink adalah "Redirect Route", memberikan deskripsi tentang tujuan dari hyperlink tersebut. Ini adalah cara yang umum digunakan untuk membuat tautan antarhalaman dalam aplikasi web Laravel yang mengarahkan pengguna kembali ke halaman utama.

Output:



Praktik Route Parameter (Required Parameter)

Buka file **routes/web.php,** praktikkan **Route Parameter (Required Parameter)** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/user/{id}', function($id) {
    return "User Id: ".$id;
});
Route::get('/posts/{post}/comments/{comment}', function($postId, $commentId) {
    return "Post Id: ".$postId.", Comment Id: ".$commentId;
});
```

Penjelasan:
Kode tersebut adalah bagian dari file routes/web.php dalam proyek Laravel. Pertama, itu mendefinisikan sebuah route dengan URL '/user/{id}'. Saat URL ini diakses, fungsi anonim yang terkait akan dijalankan, yang mengembalikan string yang berisi nilai dari parameter 'id'. Selanjutnya, ada route dengan URL '/posts/{post}/comments/{comment}'. Saat URL ini diakses, fungsi anonim yang terkait akan dijalankan, yang mengembalikan string yang berisi nilai dari parameter 'post' dan 'comment'. Dengan menggunakan parameter dinamis dalam URL, Laravel memungkinkan untuk menangani permintaan dengan lebih dinamis, seperti menampilkan informasi tentang pengguna tertentu atau menampilkan komentar dari suatu posting tertentu.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/user/12345') }}" class="list-group-item list-group-item-
action">
                Route Parameter (Required Parameter) - 1
            </a>
            <a href="{{ url('/posts/01/comments/20') }}" class="list-group-item
list-group-item-action">
                Route Parameter (Required Parameter) - 2
            </a>
```

Penjelasan:
Kode tersebut adalah bagian dari sebuah tampilan Blade dalam proyek Laravel yang menghasilkan hyperlink. Ketika hyperlink ini diklik, pengguna akan diarahkan ke URL yang sesuai. Pada hyperlink pertama, URL '/user/12345' dituju, yang mencakup parameter dinamis '12345'. Pada hyperlink kedua, URL '/posts/01/comments/20' dituju, yang mencakup parameter dinamis '01' untuk 'post' dan '20' untuk 'comment'. Ini adalah cara yang umum digunakan dalam aplikasi web Laravel untuk membuat tautan yang memanfaatkan parameter rute untuk membawa informasi tertentu ke halaman yang dituju.

Output:

Praktik Router Prameter (Optimal Parameter)

Buka file **routes/web.php,** praktikkan **Route Parameter (Optional Parameter)** dengan menuliskan kode program seperti di bawah ini.

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

Penjelasan:
Kode tersebut adalah bagian dari file routes/web.php dalam proyek Laravel. Ini mendefinisikan sebuah route dengan URL 'username/{name?}'. Tanda tanya (?) setelah parameter 'name' menandakan bahwa parameter tersebut bersifat opsional. Saat URL ini diakses, fungsi anonim yang terkait akan dijalankan. Fungsi ini mengembalikan string yang berisi nilai dari parameter 'name'. Jika tidak ada nilai yang diberikan untuk 'name', maka akan mengembalikan string kosong.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/username') }}" class="list-group-item list-group-item-action">
                Route Parameter (Optional Parameter)
            </a>
```

Penjelasan:
Kode tersebut adalah bagian dari sebuah tampilan Blade dalam proyek Laravel yang menghasilkan hyperlink. Ketika hyperlink ini diklik, pengguna akan diarahkan ke URL '/username', yang merupakan bagian dari sebuah route dengan parameter opsional 'name'. Ini memungkinkan pengguna untuk mengakses halaman dengan atau tanpa menyertakan nilai untuk parameter 'name'. Penggunaan parameter opsional memberikan fleksibilitas dalam menangani permintaan, di mana pengguna dapat memasukkan nilai untuk parameter tersebut sesuai kebutuhan.

Output:

Praktik Route With Regular ExpressionContraints

Buka file **routes/web.php,** praktikkan **Route With Regular Expression Constraints** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/title/{title}', function($title) {
    return "Title: ".$title;
})->where('title', '[A-Za-z]+');
```

Penjelasan:

Kode tersebut adalah bagian dari file routes/web.php dalam proyek Laravel. Ini mendefinisikan sebuah route dengan URL '/title/{title}'. Saat URL ini diakses, fungsi anonim yang terkait akan dijalankan, yang mengembalikan string yang berisi nilai dari parameter 'title'. Terdapat juga method where() yang mengatur constraint (pembatasan) pada parameter 'title'. Dalam kasus ini, constraint '[A-Za-z]+' mengharuskan nilai parameter 'title' hanya berisi karakter alfabet, baik huruf besar maupun huruf kecil. Dengan menggunakan constraint seperti ini, Laravel memastikan bahwa route hanya akan cocok dengan URL yang sesuai dengan pembatasan yang diberikan.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/title/this-is-my-title') }}" class="list-group-item list-group-item-action">
                Route With Regular Expression Constraints
            </a>
```

Penjelasan:
Kode di atas adalah bagian dari sebuah tampilan Blade dalam proyek Laravel yang menghasilkan hyperlink. Ketika hyperlink ini diklik, pengguna akan diarahkan ke URL '/title/this-is-my-title', yang sesuai dengan route yang memiliki regular expression constraints. Dalam kasus ini, constraint tersebut memastikan bahwa nilai parameter 'title' hanya dapat berisi karakter alfabet, angka, tanda hubung (-), dan garis bawah (_). Ini adalah cara yang umum digunakan dalam aplikasi web Laravel untuk memastikan keamanan dan kecocokan parameter dengan pola tertentu.

Output:

Praktik Named Route

Buka file **routes/web.php,** praktikkan **Named Route** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/profile/{profileId}', [RouteController::class, 'profile'])-
>name('profileRouteName');
```

Penjelasan:

Kode di atas adalah bagian dari file routes/web.php dalam proyek Laravel. Ini mendefinisikan sebuah route dengan URL '/profile/{profileId}'. Saat URL ini diakses, Laravel akan mengarahkan permintaan tersebut ke method 'profile' dalam controller 'RouteController'. Route ini juga diberi nama 'profileRouteName' menggunakan method name(). Dengan memberi nama pada route, Anda dapat dengan mudah mengacu ke route ini dari kode lain dalam proyek Laravel, misalnya saat membuat hyperlink atau melakukan pengalihan rute dalam aplikasi.

Tambahkan function bernama **“profile”** pada class **RouteController**.

```
public function profile($profileId) {
        return "This is Profile from Controller, profile id: ".$profileId;
    }
```

Penjelasan:
Kode di atas merupakan implementasi dari method 'profile' dalam controller 'RouteController' dalam proyek Laravel. Method ini menerima parameter '$profileId' dan mengembalikan sebuah string yang mencakup nilai dari parameter tersebut. Dengan cara ini, saat route '/profile/{profileId}' diakses, method 'profile' dalam controller akan dijalankan, dan pengguna akan melihat pesan yang berisi informasi tentang profil yang sesuai dengan 'profileId' yang diberikan. Ini adalah cara yang umum digunakan untuk

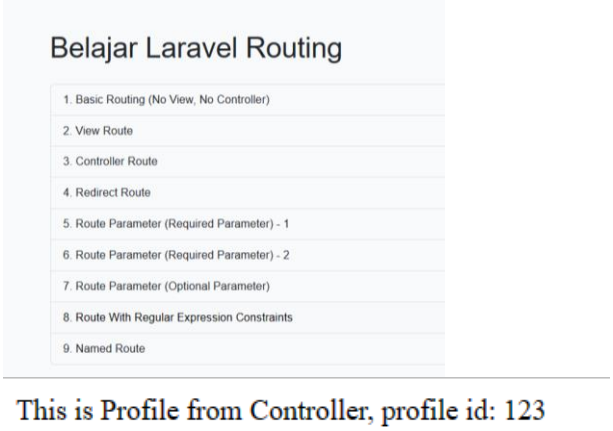
mengelola logika aplikasi dalam Laravel dan memberikan respons yang sesuai kepada pengguna.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ route('profileRouteName', ['profileId' => '123']) }}" class="list-group-item list-group-item-action">
                Named Route
            </a>
```

Penjelasan:
Kode di atas adalah bagian dari sebuah tampilan Blade dalam proyek Laravel yang menghasilkan hyperlink. Ketika hyperlink ini diklik, pengguna akan diarahkan ke route yang memiliki nama 'profileRouteName' dengan parameter 'profileId' yang bernilai '123'. Penggunaan metode route() memungkinkan untuk merujuk langsung ke sebuah route dengan nama yang telah ditentukan sebelumnya dalam file routes/web.php. Ini adalah cara yang umum digunakan dalam aplikasi web Laravel untuk membuat tautan yang mengarah ke route tertentu dengan parameter yang ditentukan.

Output:



Praktik Route Priority

Buka file **routes/web.php,** praktikkan **Route Priority** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/route_priority/{rpId}', function($rpId) {
    return "This is Route One";
});
Route::get('/route_priority/user', function() {
    return "This is Route 1";
});
Route::get('/route_priority/user', function() {
```

```
    return "This is Route 2";
});
```

Penjelasan:
Kode di atas adalah bagian dari file routes/web.php dalam proyek Laravel. Pertama, itu mendefinisikan sebuah route dengan URL '/route_priority/{rpId}'. Saat URL ini diakses, Laravel akan menjalankan function anonim yang terkait, yang mengembalikan string "This is Route One". Kemudian, ada dua route lainnya dengan URL '/route_priority/user'. Namun, karena keduanya memiliki URL yang sama, hanya route yang didefinisikan pertama kali yang akan diakses, yaitu route yang mengembalikan string "This is Route 1". Dengan demikian, route kedua yang mengembalikan string "This is Route 2" akan diabaikan karena route pertama dengan URL yang sama telah didefinisikan sebelumnya. Ini menunjukkan bahwa dalam Laravel, route akan dipetakan berdasarkan urutan definisi, dan hanya route pertama yang cocok dengan URL yang akan dieksekusi.

Comment Route yang pertama di atas, seperti kode program di bawah ini

```
// Route::get('/route_priority/{rpId}', function($rpId) {
//     return "This is Route One";
// });
Route::get('/route_priority/user', function() {
    return "This is Route 1";
});
Route::get('/route_priority/user', function() {
    return "This is Route 2";
});
```

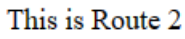
Penjelasan:
Komentar pada route pertama menunjukkan bahwa route tersebut dinonaktifkan, sehingga tidak akan dipertimbangkan oleh Laravel dalam penentuan rute yang akan dieksekusi.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/route_priority/user') }}" class="list-group-item list-group-item-action">
                Route Priority
            </a>
```

Penjelasan:
Kode di atas adalah bagian dari sebuah tampilan Blade dalam proyek Laravel yang menghasilkan hyperlink. Ketika hyperlink ini diklik, pengguna akan diarahkan ke URL '/route_priority/user'. Ini adalah cara yang umum digunakan dalam aplikasi web Laravel untuk membuat tautan antarhalaman. Dalam hal ini, hyperlink ini mengarah ke sebuah route yang akan menampilkan halaman dengan prioritas tertentu, yang dapat ditentukan dalam file routes/web.php.

Output:



Praktik Fallback Router

Buka file **routes/web.php,** praktikkan **Fallback Routes** dengan menuliskan kode program seperti di bawah ini.

```
Route::fallback(function() {
    return 'This is Fallback Route';
});
```

Penjelasan:
Kode di atas adalah bagian dari file routes/web.php dalam proyek Laravel. Ini mendefinisikan sebuah fallback route, yang akan dijalankan jika tidak ada route lain yang cocok dengan URL yang diminta oleh pengguna. Fallback route sering digunakan untuk menangani permintaan untuk URL yang tidak ditangani oleh route lain dalam aplikasi. Dalam contoh ini, saat pengguna mengakses URL yang tidak sesuai dengan route yang didefinisikan sebelumnya, Laravel akan menjalankan fungsi anonim yang terkait dengan fallback route dan mengembalikan pesan "This is Fallback Route". Ini memberikan perlindungan terhadap kemungkinan kesalahan atau URL yang tidak valid dalam aplikasi Laravel.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/asdqwezxc') }}" class="list-group-item list-group-item-action">
                Fallback Routes
            </a>
```

Penjelasan:
Kode di atas adalah bagian dari sebuah tampilan Blade dalam proyek Laravel yang menghasilkan hyperlink. Ketika hyperlink ini diklik, pengguna akan diarahkan ke URL '/asdqwezxc'. Dalam konteks "Fallback Routes", hal ini dapat diartikan bahwa jika tidak ada route yang cocok dengan URL yang diminta oleh pengguna (dalam hal ini '/asdqwezxc'), maka Laravel akan menjalankan fallback route yang telah ditentukan

sebelumnya. Ini adalah cara yang umum digunakan dalam aplikasi web Laravel untuk menangani permintaan untuk URL yang tidak sesuai dengan route yang telah didefinisikan sebelumnya.

Output:

This is Fallback Route

Praktik Route Groups (Route Prefixes & Route Name Prefixes)

Buka file **routes/web.php,** praktikkan **Route Groups (Route Prefixes & Route Name Prefixes)** dengan menuliskan kode program seperti di bawah ini.

```
Route::prefix('admin')->name('admin.')->group(function() {
    Route::get('/dashboard', function() {
        return "This is admin dashboard";
    })->name('dashboard');
    Route::get('/users', function() {
        return "This is user data on admin page";
    })->name('users');
    Route::get('/items', function() {
        return "This is item data on admin page";
    })->name('items');
});
```

Penjelasan:

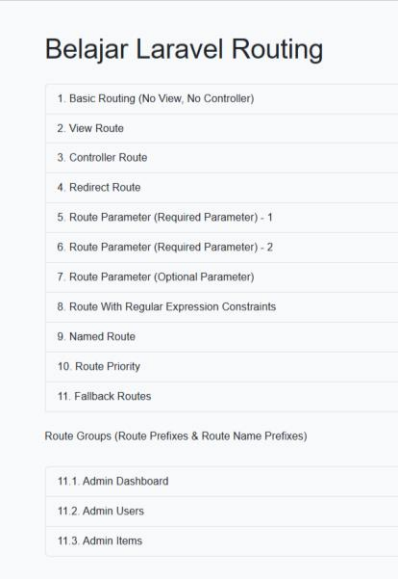
Kode tersebut mengelompokkan route dengan awalan 'admin' dan memberi nama setiap route dengan awalan 'admin.'. Semua route dalam grup ini memiliki awalan URL '/admin' dan memiliki fungsi anonim yang akan dijalankan saat diakses. Ini memudahkan dalam pengorganisasian dan pengelolaan route dalam aplikasi Laravel, terutama untuk bagian-bagian tertentu seperti halaman admin.

Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<h6 class="mt-4">Route Groups (Route Prefixes & Route Name Prefixes)</h6>
            <div class="list-group list-group-numbered mt-4">
            <a href="{{ route('admin.dashboard') }}" class="list-group-item list-
group-item-action">
```

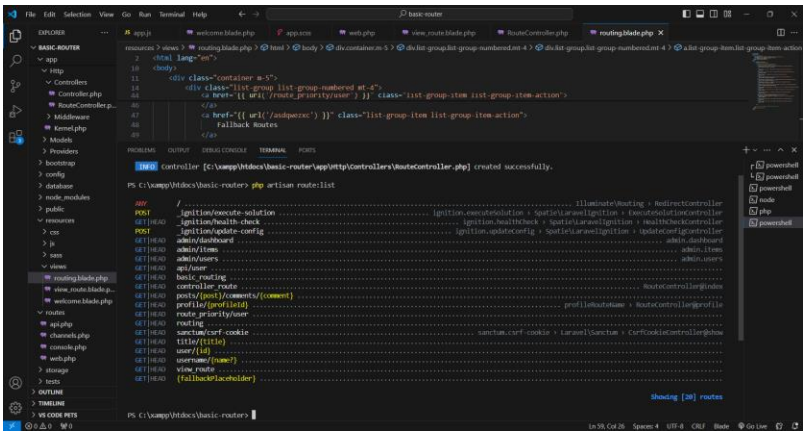
```
                Admin Dashboard
            </a>
            <a href="{{ route('admin.users') }}" class="list-group-item list-
group-item-action">
                Admin Users
            </a>
            <a href="{{ route('admin.items') }}" class="list-group-item list-
group-item-action">
                Admin Items
            </a>
```

Penjelasan:
Kode di atas adalah bagian dari sebuah tampilan Blade dalam proyek Laravel yang menampilkan hyperlink untuk navigasi. Ketika hyperlink ini diklik, pengguna akan diarahkan ke route yang sesuai dengan nama yang telah ditentukan sebelumnya. Hyperlink ini terkait dengan route grup yang memiliki awalan 'admin', sehingga setiap route memiliki awalan 'admin.'. Ini memudahkan dalam pembuatan hyperlink atau penggunaan route dalam kode lain dalam proyek Laravel, terutama untuk bagian-bagian tertentu seperti halaman admin.
Output:



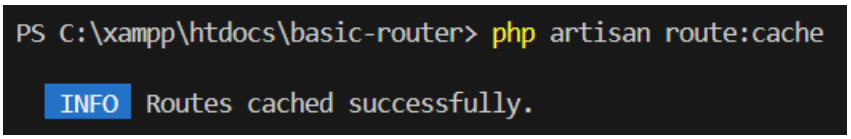
Praktik View Route List

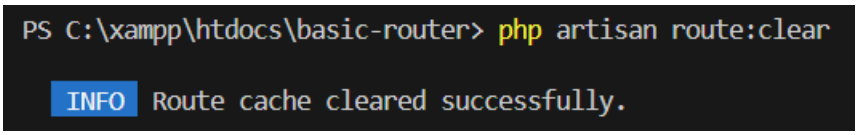
Ketikkan pada cmd / terminal script artisan berikut ini: **php artisan route:list**



Penjelasan:
Perintah php artisan route:list digunakan dalam terminal atau command prompt untuk menampilkan daftar semua route yang telah didefinisikan dalam proyek Laravel. Output dari perintah ini akan menampilkan informasi seperti metode HTTP, URI, nama route, serta controller dan method yang menanganinya. Ini sangat berguna untuk memeriksa semua route yang tersedia dalam proyek Laravel, serta untuk memastikan bahwa route telah didefinisikan dengan benar.

Praktik Route Caching





Penjelasan:
Perintah php artisan route:cache digunakan dalam terminal atau command prompt untuk menerapkan caching pada semua route yang telah didefinisikan dalam proyek Laravel. Ini membantu meningkatkan performa aplikasi dengan mempercepat proses pencarian route saat aplikasi dijalankan.

Sementara itu, perintah php artisan route:clear digunakan untuk menghapus cache dari semua route yang telah didefinisikan sebelumnya dalam proyek Laravel. Ini berguna ketika ada perubahan dalam definisi route atau ketika ingin menghapus cache route untuk alasan tertentu, seperti ketika melakukan pengembangan atau debugging.

Tugas 1

View personal web

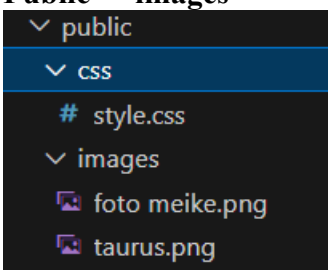
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>personal web meike</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha2/dist/css/bootstrap.min.
css">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.6.0/font/bootstrap-icons.css
">

    <link rel="stylesheet" href="../css/style.css">
</head>
<body>
    <nav class="navbar navbar-light bg-light">
        <div class="container">
          <a class="navbar-brand" href="#">
            <img src="images/taurus.png" alt="" width="50" height="35">
          </a>
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
              <li class="nav-item">
                <a class="nav-link active" aria-current="page" href="#">Home</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="#">About</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="#">Portofolio</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="#">Blog</a>
              </li>
              <li class="nav-item">
```

```
                <a class="nav-link" href="#">Contact</a>
              </li>
            </ul>
          </div>
        </div>
      </nav>

    <div class="bg-light mt-5" id="main">
      <!-- Container -->
      <div class="container py-5 px-4">
          <div class="row">
              <div class="col-md-5 order-md-2">
                  <img class="img-fluid" src="images/foto meike.png" alt="main
logo">
              </div>
              <div class="col-md-7 order-md-1">
                  <h1 class="mt-4 display-3"><b> Hello I'm Meike!!</b></h1>
                  <p class="fs-5 mt-3">I am someone who likes to learn new things
and is
                    stubborn of course because I am the last child. Has a story-
telling nature but is introverted.
                    And that's me.I am a taurus known as a very tough and
persistent individual. They don't like rapid changes and prefer to have a
structured routine.
                    They are deeply attached to traditional values and have a
strong drive to create stability in their lives, be it in terms of finances,
relationships, or careers.</p>
                  <div class="row">
                      <div class="d-flex flex-column flex-md-row">
                        <button type="button" class="btn btn-
secondary">Introduction</button>
                      </div>
                  </div>
              </div>
          </div>
      </div>
  </div>

  <div id="footer" class="bg-light py-5">
    <!-- Container -->
    <div class="container py-5 px-4">
        <div class="row">
            <div class="col-lg-3 mb-5">
                <a href="#" class="logo text-decoration-none">
                    <div class="d-flex">
```

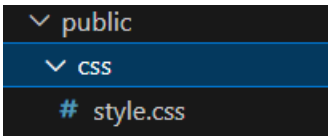
```
                        <img class="img-fluid" src="images/taurus.png"
alt="Bootstrap Logo" style="width: 40px;">
                        <div class="fs-5 ms-2 text-black">Taurus</div>
                    </div>
                </a>
                <div class="mt-2 text-muted">
                    <small>a zodiac runs from April 20 to May 20. People born
between these dates are considered to have the zodiac sign Taurus.<a href="#"
class="text-black">Taurus Team</a> with the help of
                    <a href="#" class="text-black">our contributors</a>.</small>
                </div>
                <div class="mt-2 text-muted">
                    <small>Code licensed <a href="#" class="text-black">MIT</a>,
docs <a href="#" class="text-black">CC BY 3.0</a>.</small>
                </div>
                <div class="mt-2 text-muted">
                    <small>Currently v5.3.0-alpha2.</small>
                </div>
            </div>
            <div class="col-6 col-lg-2 offset-lg-1 mb-5">
            <h5>Links</h5>
            <ul class="list-unstyled">
                <li class="mb-2"><a href="#">Home</a></li>
                <li class="mb-2"><a href="#">About</a></li>
                <li class="mb-2"><a href="#">Portofolio</a></li>
                <li class="mb-2"><a href="#">Blog</a></li>
                <li class="mb-2"><a href="#">Contact</a></li>
            </ul>
        </div>
        <div class="col-6 col-lg-2 mb-5">
            <h5>Guides</h5>
            <ul class="list-unstyled">
                <li class="mb-2"><a href="#">Getting started</a></li>
                <li class="mb-2"><a href="#">Starter template</a></li>
                <li class="mb-2"><a href="#">Webpack</a></li>
                <li class="mb-2"><a href="#">Parcel</a></li>
            </ul>
        </div>
        <div class="col-6 col-lg-2 mb-5">
            <h5>Projects</h5>
            <ul class="list-unstyled">
                <li class="mb-2"><a href="#">Bootstrap 5</a></li>
                <li class="mb-2"><a href="#">Bootstrap 4</a></li>
                <li class="mb-2"><a href="#">Icons</a></li>
                <li class="mb-2"><a href="#">RFS</a></li>
```

```
                <li class="mb-2"><a href="#">npm starter</a></li>
            </ul>
        </div>
        <div class="col-6 col-lg-2 mb-5">
            <h5>Community</h5>
            <ul class="list-unstyled">
                <li class="mb-2"><a href="#">Issues</a></li>
                <li class="mb-2"><a href="#">Discussions</a></li>
                <li class="mb-2"><a href="#">Corporate sponsors</a></li>
                <li class="mb-2"><a href="#">Open Collective</a></li>
                <li class="mb-2"><a href="#">Slack</a></li>
                <li class="mb-2"><a href="#">Stack overflow</a></li>
            </ul>
        </div>
    </div>
</div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha2/dist/js/bootstrap.bundle
.min.js"></script>
</body>
</html
```

Public -> images



Public -> css



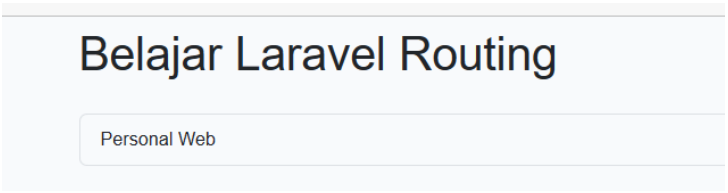
View -> routing.blade.php

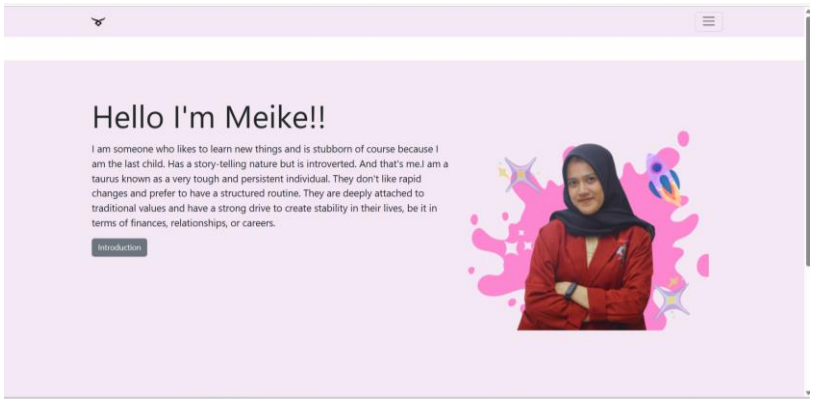
```
<div class="list-group mt-4">
            <a href="{{ url('/meike') }}" class="list-group-item list-group-item-
action">
                Personal Web
            </a>
        </div>
```

Route -> web.php

```
Route::get('/meike', function() {
    return view('meike');
});
```

Output:





Penjelasan:
Kode pertama menempatkan code website pada bagian view dengan nama file meike.blade.php dan melakukan perpindahan file images dan css pada public lalu memasukkan routing.blade.php berisi kode di atas adalah bagian dari sebuah tampilan HTML yang menggunakan Bootstrap. Ini menampilkan sebuah hyperlink dalam bentuk daftar dengan gaya yang telah ditentukan oleh Bootstrap. Saat hyperlink ini diklik, pengguna akan diarahkan ke URL '/meike'. Teks di dalam hyperlink adalah "Personal Web", yang memberikan deskripsi tentang tujuan dari hyperlink tersebut. Ini adalah cara yang umum digunakan dalam pembuatan tautan antarhalaman dalam sebuah halaman web. Kode di atas adalah bagian dari file routes/web.php dalam proyek Laravel. Ini mendefinisikan sebuah route dengan URL '/meike'. Saat URL ini diakses, Laravel akan memanggil function anonim yang terkait, yang mengembalikan sebuah view dengan nama 'meike'. Dengan cara ini, saat URL '/meike' diakses melalui browser, pengguna akan melihat tampilan (view) yang telah ditentukan dalam file 'meike.blade.php' di dalam direktori resources/views. Ini adalah cara yang umum digunakan dalam Laravel untuk menampilkan halaman web kepada pengguna berdasarkan rute tertentu.