

LARAVEL ROUTING

KELAS PROGRAMMING FULL STACK DEVELOPER

MITRA PELATIHAN



Jabar Digital Academy

digitalacademy.jabarprov.go.id

2024

BAB II

LARAVEL ROUTING

1. Tujuan

- a. Peserta didik dapat mengetahui serta kegunaan *Routing*.
- b. Peserta didik dapat memahami cara kerja *Routing*.
- c. Peserta didik dapat membuat *route*.
- d. Peserta didik dapat membuat route dengan path URL yang dinamis.
- e. Peserta didik dapat memberikan nama pada *route*.
- f. Peserta didik dapat mengelompokkan *route* yang dibuat.

2. Perlengkapan

- a. Modul 2. LARAVEL ROUTING.
- b. IDE atau Teks Editor (Visual Studio Code, Notepad++, Sublime)

3. Materi

Bayangkan Anda memiliki peta kota yang besar dengan berbagai jalan dan persimpangan. Setiap jalan mewakili rute yang dapat diambil oleh pengguna untuk mencapai tujuan mereka.

Dalam Laravel, sistem routing bertindak seperti peta kota tersebut. Setiap rute dalam sistem routing adalah jalan yang menghubungkan antara URL yang diminta oleh pengguna dengan aksi yang akan dilakukan oleh aplikasi web Anda. Misalnya, ketika pengguna mengunjungi URL tertentu, sistem routing akan menentukan rute yang sesuai dan menetapkan aksi yang harus dilakukan, seperti menampilkan halaman tertentu atau memproses data yang dikirim oleh pengguna.

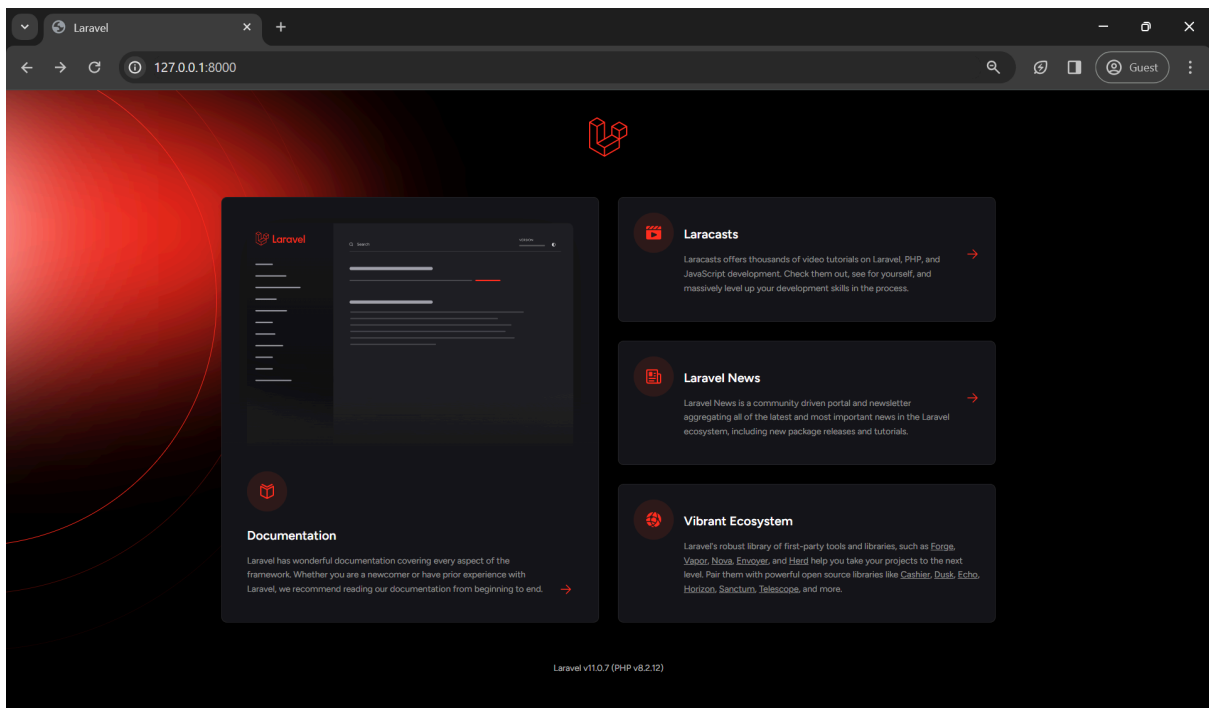
1) Apa Itu Routing

Routing adalah salah satu komponen yang digunakan untuk mendefinisikan alamat URL, setiap URL memiliki perilakunya masing masing yang akan kita atur di dalam controller pada aplikasi kita. Dengan Routing kita dapat membuat sebuah struktur

url yang mudah dikelola jika aplikasi sudah kompleks, untuk melihat dimana kita dapat mengelola Route. Untuk membuat route kita perlu memanggil class bernama Route lalu diikuti dengan methodnya, argument pertama diisi dengan nilai URL dan argumen kedua diisi dengan logic yang akan memproses permintaan user. *Code* membuat Route seperti ini :

```
Route::get(uri, function(){ logic })
```

Ketika pertama kali kita jalankan laravel di browser maka akan yang akan ditampilkan oleh browser adalah tampilan seperti ini.



Tampilan Awal Laravel.

Tampilan ini berasal dari routing yang ada di dalam folder *router/web.php*.

```
<?php
// routes/web.php
use Illuminate\Support\Facades\Route;

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

Route Bawaan Untuk Menampilkan Halaman Utama.

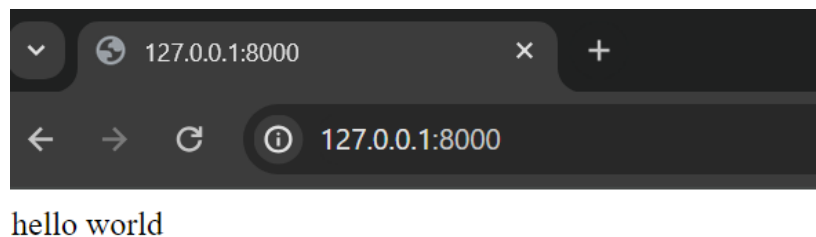
Didalam route berisikan route bawaan yang didalamnya menampilkan file yang ada didalam folder *resources/views/welcome.blade.php* seperti itu. Semua rute Laravel ditentukan dalam file rute Anda, yang terletak di direktori routes. File - file ini secara otomatis dimuat oleh file *bootstrap/app.php* kita dapat mengatur didalamnya jika diperlukan. kita akan coba return diatas kita ubah menjadi *hello world*.

```
// routes/web.php

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

Nilai Return Yang Diubah.

Ketika kita jalankan di browser maka akan tampil seperti ini.



Tampilan Setelah Nilai Return Diubah.

Jika kita perhatikan kita mendefinisikan nilai route “/” namun di browser character “/” tidak terlihat di url, itu dikarenakan character “/” digunakan untuk memisahkan bagian yang ada di url, jika setelah “/” tidak ada *character* atau string yang mengikuti, maka “/” tidak akan ditampilkan di url.

Router memungkinkan Anda untuk mendaftarkan rute yang merespons kata kerja HTTP apapun. Berikut ada beberapa method yang biasa digunakan pada http method, diantaranya adalah sebagai berikut.

- GET
Method GET meminta representasi sumber daya yang ditentukan. Permintaan menggunakan GET seharusnya hanya mengambil data.
- POST
Method POST digunakan untuk mengirimkan entitas ke sumber daya yang ditentukan, sering menyebabkan perubahan pada keadaan atau efek samping pada server.
- PUT
Method PUT menggantikan semua representasi terkini dari sumber target dengan muatan permintaan.
- PATCH
Method PATCH digunakan untuk menerapkan modifikasi sebagian pada sumber daya.
- DELETE
Method DELETE digunakan untuk menghapus sumber daya yang ditentukan.
- ANY
Route yang menggunakan Method ANY dapat memanggil dengan method diatas.

```
Route::get(URL, callback);  
Route::post(URL, callback);  
Route::put(URL, callback);  
Route::patch(URL, callback);  
Route::delete(URL, callback);  
Route::any(URL, callback);
```

Method Didalam Route.

Kita dapat menggunakan function *resources* yang ada di dalam class Route. dengan menggunakan *resources* kita mendefinisikan *get*, *post*, *put*, *patch*, *delete* dan *any* dijadikan satu baris *code*, namun untuk menggunakan *function* ini kita perlu membuat controller terlebih dahulu.

2) Passing Data Di URL.

Terkadang kita ingin mengirim nilai ke dalam logic yang ada didalam route, misalnya kita ingin menampilkan data user yang membutuhkan data usernya, kita bisa mengirim data dengan menyisipkan data data tersebut di dalam di url.

Kita akan mencoba membuat route lagi dan beri nama `"/users"` yang berisikan array data *user*, seperti ini.

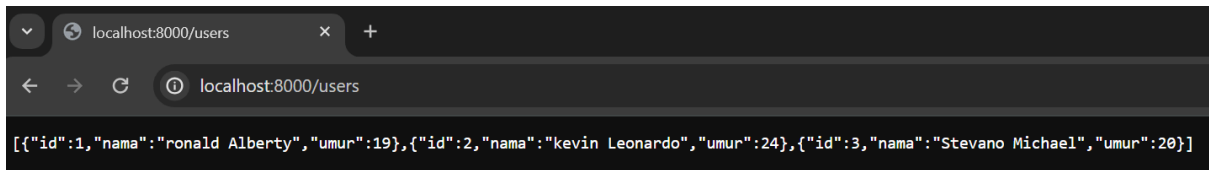
```
// routes/web.php

Route::get("/users", function(){
    $users = [
        [
            "id" => 1,
            "nama" => "ronald Alberty",
            "age" => 20,
        ],
        [
            "id" => 2,
            "nama" => "kevin Leonardo",
            "age" => 24,
        ],
        [
            "id" => 3,
            "nama" => "Stevano Michael",
            "age" => 20,
        ]
    ];

    return $users;
});
```

Code Route Users.

Jika kita jalankan di browser maka akan terlihat seperti ini



Hasil Dari Route.

Kita telah berhasil menampilkan semua data yang ada didalam array. Namun bagaimana cara menampilkan data tertentu saja, misalnya data yang ditampilkan itu yang memiliki ID tertentu saja. Kita dapat melakukan itu dengan melakukan for looping lalu beri kondisi jika id nya memenuhi kondisi maka tampilkan, untuk sementara kita atur nilai di dalam kondisi *if* berupa nilai statis baris code nya seperti ini.

```
// routes/web.php

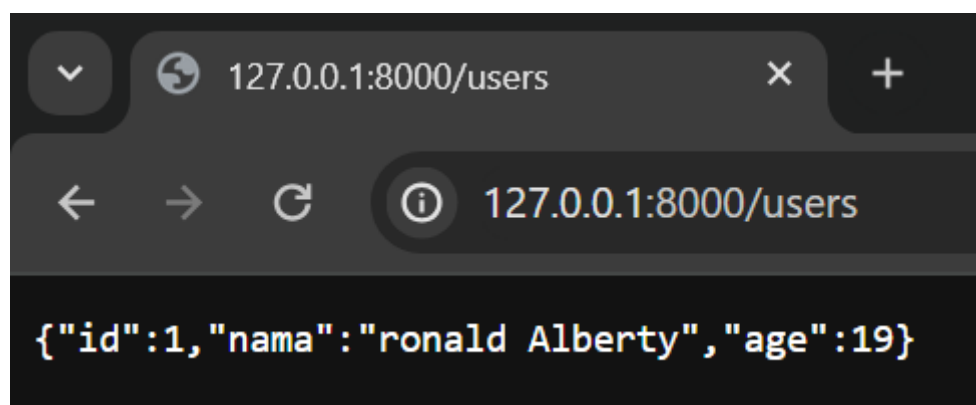
Route::get("/users", function(){
    $users = [
        [
            "id" => 1,
            "nama" => "ronald Alberty",
            "umur" => 20,
        ],
        [
            "id" => 2,
            "nama" => "kevin Leonardo",
            "umur" => 24,
        ],
        [
            "id" => 3,
            "nama" => "Stevano Michael",
            "umur" => 20,
        ]
    ];

    $result = [];
    foreach($users as $user){
        if($user["id"] == 1){
            array_push($result, $user);
        }
    }

    return $result;
});
```

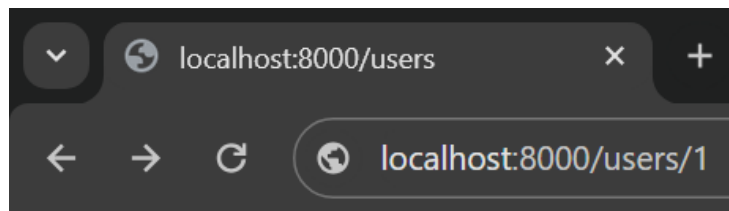
Penambahan *Foreach* dan *Conditional*.

Ketika kita jalankan, maka menampilkan data seperti ini.



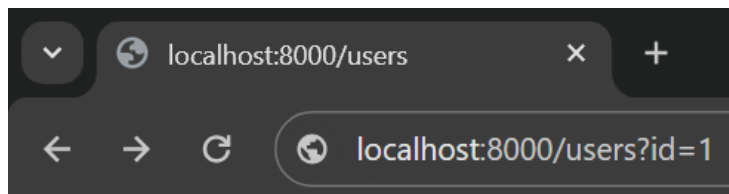
Tampilan Data Yang Telah Di Sorting.

Jika kita perhatikan di dalam logic nilai yang diberikan ke dalam *routing* sifatnya *statis* atau tidak dinamis, jadi kalau kita ingin merubah *user* yang akan ditampilkan kita perlu ubah secara manual dalam kondisi *if* seperti diatas. Kita dapat mengirim data yang dikirim dari URL dan menggunakannya, ada beberapa cara untuk mengirim data lewat URL, mulai dari menyisipkan data di dalam path atau membuat parameter di baris url, contoh URL yang disisipi data seperti ini.



Contoh Url Yang Disisipi Data Didalamnya.

Adapun dengan membuat query url seperti ini.



Contoh Url Yang Disisipi Data Didalamnya.

Query url adalah bagian url yang digunakan untuk menyisipkan data tambahan. Query URL dimulai setelah tanda tanya (?) dan terdiri dari satu atau lebih pasangan antara key dan value yang dipisahkan oleh tanda tambah (=), dan ampersand (&) untuk memberikan data.

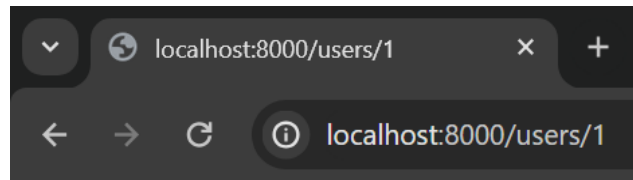
Agar path di baris URL menjadi dinamis, saat kita membuat route kita bisa tambahkan kurung kurawal buka lalu di dalamnya diisi dengan nama nilai yang akan disisipkan lalu di dalam function beri nama variabel yang sama dari url yang diberikan , kita langsung praktekkan saja dengan membuat route baru seperti ini.

```
// routes/web.php

Route::get('/users/{id}', function ($id) {
    var_dump($id);
});
```

Pembuatan Route Dengan Url Dinamis.

Ketika kita coba akses di browser maka akan menampilkan id yang kita kirim di url, tipe data dari nilai yang dikirim dari URL yaitu string. Ketika kita jalankan maka akan tampil seperti ini.



string(1) "1"

Data Yang Dikirim Dari URL.

Kita akan coba filter data user yang ada didalam route `"/users"` berdasarkan ID yang dikirim lewat URL, kita ubah nilai yang ada di dalam kondisi if sebelumnya *statis* menjadi id yang dikirim dari URL. Contoh kodenya seperti dibawah ini.

```
// routes/web.php

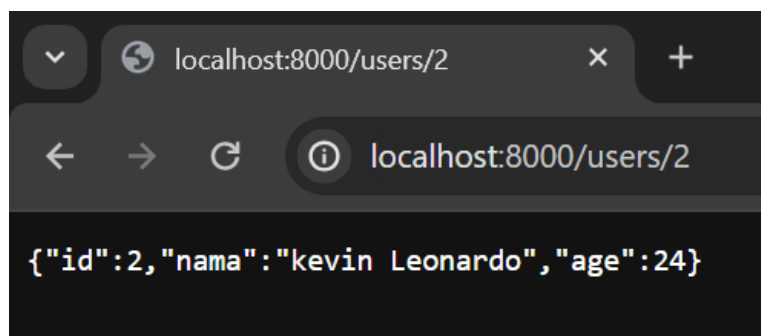
Route::get("/users/{id}", function($id){
    $users = [
        [
            "id" => 1,
            "nama" => "ronald Alberty",
            "age" => 20,
        ],
        [
            "id" => 2,
            "nama" => "kevin Leonardo",
            "age" => 24,
        ],
        [
            "id" => 3,
            "nama" => "Stevano Michael",
            "age" => 20,
        ]
    ];

    $result = [];
    foreach($users as $user){
        if($user["id"] == $id){
            array_push($result, $user);
        }
    }

    return $result;
});
```

Code Menampilkan User Berdasarkan ID Dari URL.

Ketika kita jalankan data yang akan ditampilkan berdasarkan ID yang dikirim dari URL.



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/users/2'. Below the address bar, the JSON response is displayed: `{"id":2,"nama":"kevin Leonardo","age":24}`.

Data User Yang Ditampilkan Berdasarkan ID Dari URL.

Kita juga dapat membuat agar path yang disisipi data bersifat tidak wajib atau optional, cukup tambahkan tanda tanya (?) setelah nama di dalam url pada saat membuat route, selain memberikan tanda tanya (?) setelah nama di dalam url perlu

kita tambahkan juga tanda tanya (?) dan tentukan juga tipe datanya, contohnya seperti ini seperti ini.

```
// routes/web.php

Route::get('/users/{id?}', function(?string $id = null){
    $users = [
        [
            "id" => 1,
            "nama" => "ronald Alberty",
            "umur" => 20,
        ],
        [
            "id" => 2,
            "nama" => "kevin Leonardo",
            "umur" => 24,
        ],
        [
            "id" => 3,
            "nama" => "Stevano Michael",
            "umur" => 20,
        ]
    ];

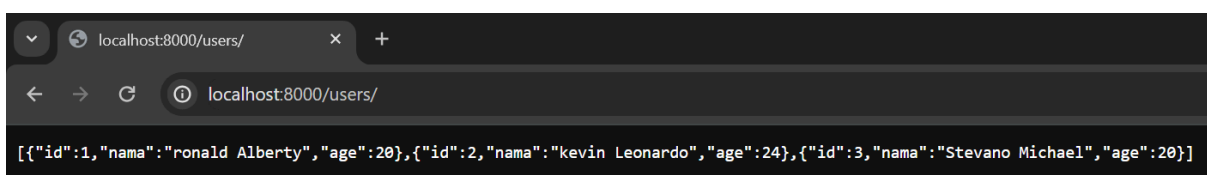
    $result = [];

    if($id != null){
        foreach($users as $user){
            if($user["id"] == $id){
                $result = $user;
            }
        }
    } else{
        $result = $users;
    }

    return $result;
})->name("users.detail");
```

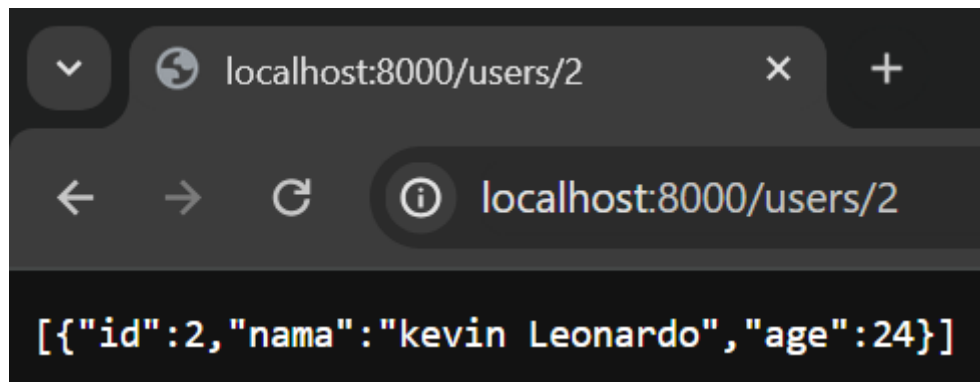
Contoh Route Yang Nilai.

dengan begini kita tidak perlu membuat 2 route baru karena dalam satu route sudah mencakup 2 aksi dengan kondisi yang berbeda, misalnya jika *path* setelah *users* itu kosong maka tampilkan semua *users*, jika *path* setelah *users* bernilai, maka tampilkan sesuai dengan ID yang diberikan dari *path* URL.



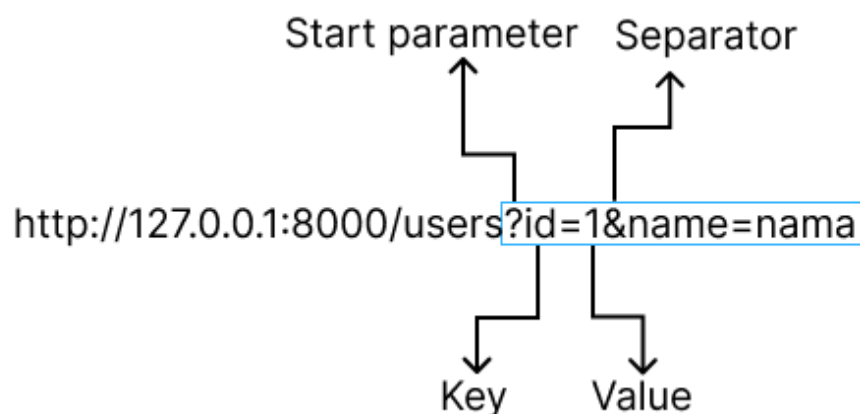
The screenshot shows a web browser window with the address bar displaying 'localhost:8000/users/'. The page content shows a JSON array of three user objects: [{"id":1,"nama":"ronald Alberty","age":20}, {"id":2,"nama":"kevin Leonardo","age":24}, {"id":3,"nama":"Stevano Michael","age":20}].

Tampilan Jika Path URL Setelah Users Kosong.



Tampilan Jika Path URL Setelah Users Tidak Kosong.

Kita sudah mencoba menggunakan path dari URL, sekarang kita akan coba menggunakan data yang ada di dalam URL dengan parameter. Untuk menggunakan nilai yang ada di dalam parameter kita perlu tambahkan tanda tanya (?) sebagai tanda mulainya parameter, lalu diikuti dengan nama parameternya dan nilainya, dan juga penggunaan parameter di dalam url sangat dianjurkan dibandingkan menyisipkannya di dalam path karena nilai yang dikirim tidak ambigu ketika data yang dikirim banyak karena ketika kita membuat parameter di url itu seperti kita membuat variabel sehingga kita dapat mendefinisikan nama variabelnya di baris URL. Kita juga dapat membuat parameter lebih dari satu, Untuk membuat Parameter di dalam URL itu seperti ini.



Contoh URL Dengan Parameter.

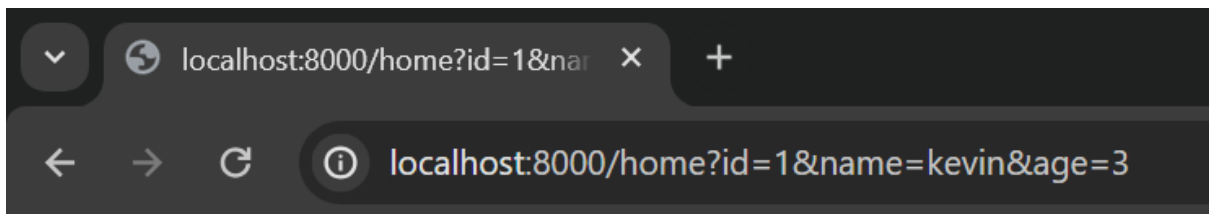
Untuk menggunakan data yang dikirim melalui parameter di dalam URL, kita perlu memanggil terlebih dahulu *class Request* yang disediakan oleh laravel. Kita akan

route baru lalu panggil *class request* sebagai parameter di dalam *callback*, seperti ini.

```
Route::get("/home", function(Request $request){});
```

Ditambahkannya *Class Request*.

Lalu di url kita tambahkan parameter bernama id, name, dan age. seperti ini.



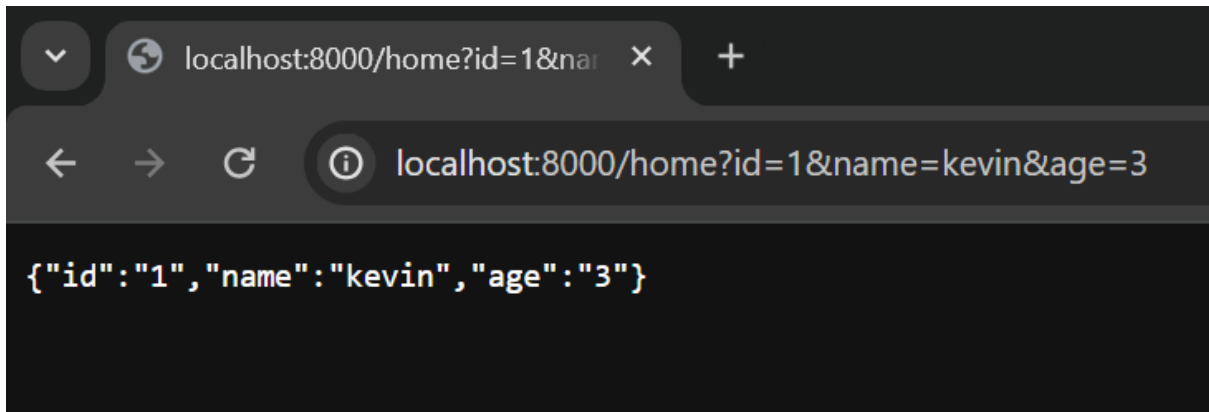
Url Yang Diberi Parameter.

Lalu didalam function kita panggil query untuk melihat apakah data yang dikirim melalui Parameter dapat digunakan atau tidak.

```
Route::get("/home", function(Request $request){  
    return $request->query( );  
});
```

Code Pemanggilan Query.

Lalu kita jalankan di dalam browser maka akan menampilkan data yang dikirimkan dari URL.



Tampilan Data Dari URL.

Jika sudah menampilkan object seperti ini berarti data yang dikirim dari URL dapat kita gunakan.

Class Request mengatur segala bentuk jenis *request* atau permintaan dalam HTTP Request. Untuk menggunakan nilai yang dikirim dari url kita cukup gunakan parameter request lalu panggil objeknya berdasarkan nama parameter yang ada di URL. Kita akan langsung coba di *route "/users"*, kita check terlebih dahulu apakah ada param yang dikirim sebelum melakukan *filter* berdasarkan id, nama, dan umur yang datanya kita dapatkan dari Parameter.

```
// routes/web.php

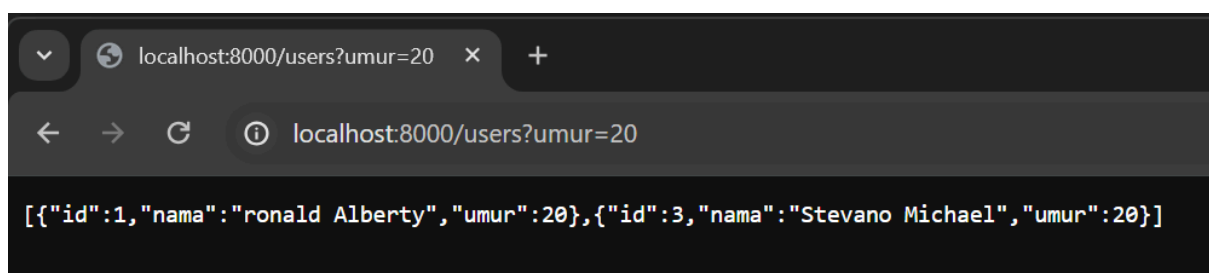
Route::get('/users', function(Request $request){
    $users = [
        [
            "id" => 1,
            "nama" => "ronald Alberty",
            "umur" => 20,
        ],
        [
            "id" => 2,
            "nama" => "kevin Leonardo",
            "umur" => 24,
        ],
        [
            "id" => 3,
            "nama" => "Stevano Michael",
            "umur" => 20,
        ]
    ];

    $result = [];
    if($request->query() != null){
        foreach($users as $user){
            if(
                $user["id"] == $request->id ||
                $user["nama"] == $request->nama ||
                $user["umur"] == $request->umur
            ){
                array_push($result, $user);
            }
        }
    } else{ $result = $users; }

    return $result;
});
```

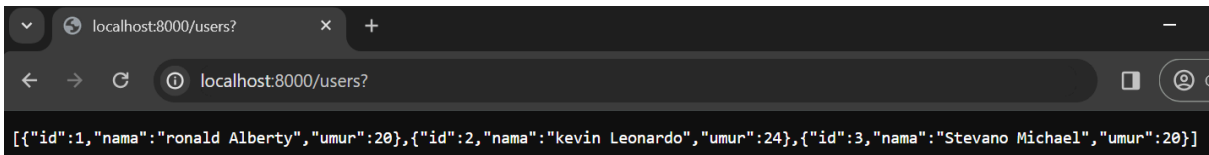
Code Yang Diberi Filter Berdasarkan Data Dari URL.

Ketika kita jalan di browser lalu kita tambahkan Params nama di dalamnya, namun kita dapat memasukkan id, dan umur seperti ini.



Penambahan Params Nama.

kita coba kosongkan param didalam URL, maka akan menampilkan semua data user.



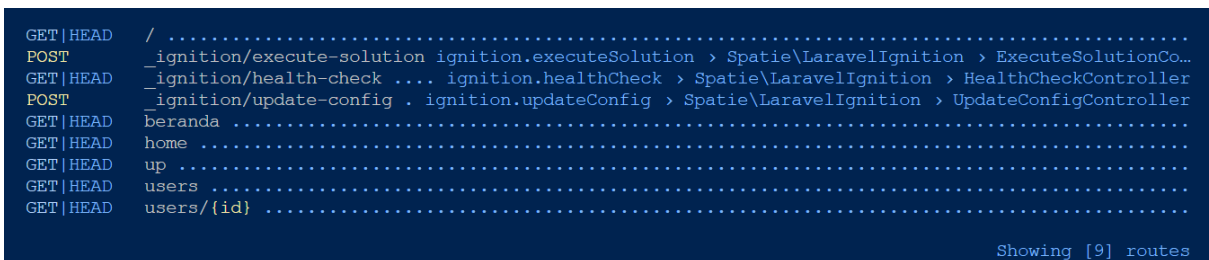
Tampilan Ketika Param Dikosongkan.

3) Named Route.

Ketika aplikasi kita sudah kompleks route yang kita buat sudah banyak dan akan kesulitan untuk manage semua itu, untuk mempermudah semua itu kita dapat melihat semua route yang ada dengan perintah ini.

php artisan route:list

Ketika kita jalankan perintah di atas di terminal maka akan muncul semua list route yang terdaftar di dalam aplikasi kita.



Tampilan Daftar Route.

Selain kita menggunakan perintah diatas, kita juga dapat memberikan label atau nama di setiap *route* yang kita buat, dengan begitu dapat memudahkan kita untuk mengelola route yang sudah banyak. Untuk menambahkan label atau nama pada route kita cukup panggil *method* atau *function* name setelah get, contohnya seperti ini.



Penamaan Route.

Keuntungan memberikan label atau nama pada *route* adalah mudah dibaca sehingga dapat mempermudah mengelola ketika *route* yang dimiliki sudah banyak, juga mudah dipahami oleh programmer lain jika kita mengembangkan aplikasi dalam sebuah tim. Kita dapat gunakan nama route ini saat kita perlu arahkan user ke suatu route contohnya codenya seperti ini.

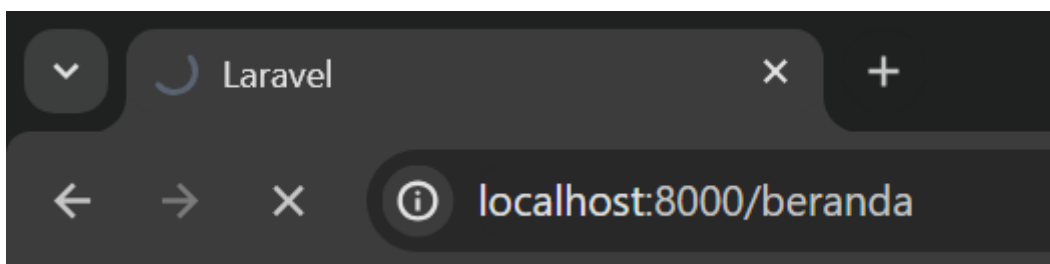
```
// routes/web.php

Route::get('/', function () {
    return view('welcome');
})->name("homepage");

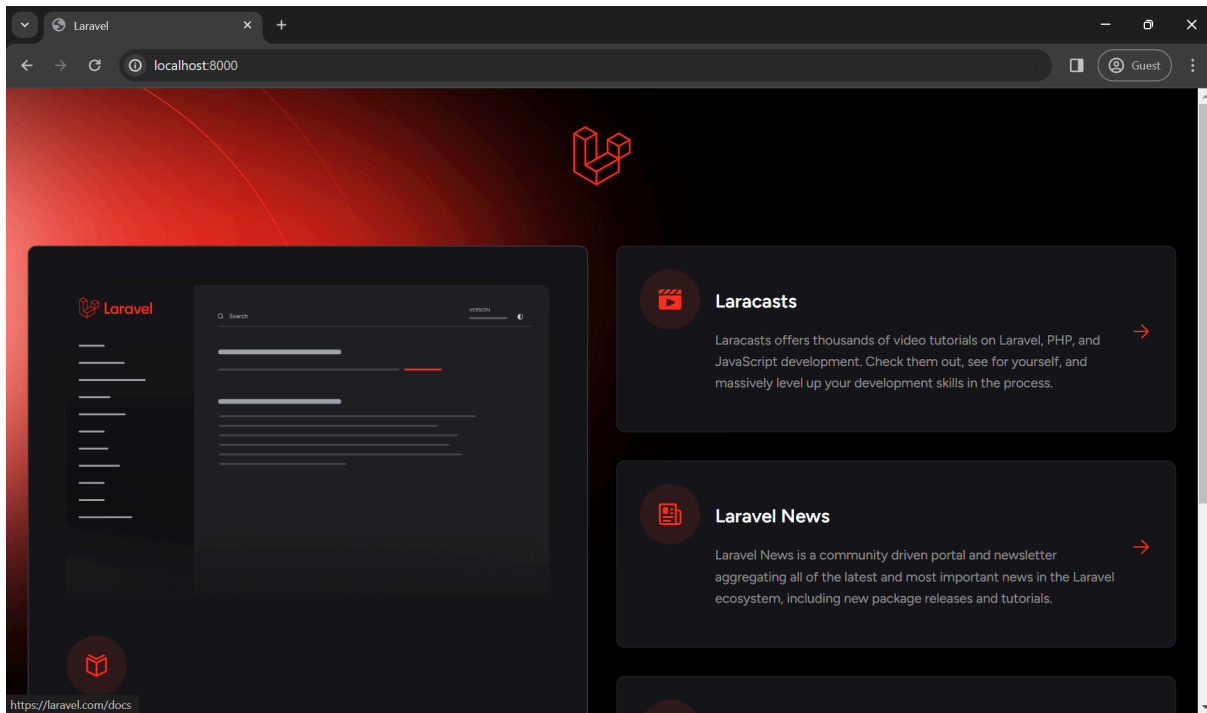
Route::get("/beranda", function(){
    sleep(5);
    return redirect()->route("homepage");
});
```

Route Untuk Mengarahkan Ke Halaman Awal.

Ketika kita akses *route* beranda maka akan terjadi loading selama 5 detik lalu akan diarahkan ke halaman awal.



Saat Akses Route Beranda.



User Akan Diarahkan Ke Halaman Home.

Ketika kita jalankan perintah `artisan route:list` maka route yang sudah diberi nama menjadi seperti ini.

```
GET|HEAD / ..... homepage
POST _ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionContro...
GET|HEAD _ignition/health-check ..... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST _ignition/update-config ..... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD beranda .....
GET|HEAD up .....
GET|HEAD users .....
GET|HEAD users/{id?} .....

Showing [8] routes
```

Tampilan Ketika Route Yang Diberi Nama.

4) Grouping Route.

Ketika kita membuat route di Laravel, seringkali kita harus menentukan beberapa hal yang sama untuk setiap route, seperti nama, URL, controller, dan middleware. Ini bisa jadi cukup merepotkan karena kita harus menuliskannya berulang kali untuk setiap *route* yang kita definisikan. Namun, dengan mengelompokkan *route*, kita dapat menghindari perulangan tersebut. Kita bisa menentukan semua pengaturan yang sama sekali di satu tempat, dan itu akan berlaku untuk semua route dalam grup tersebut. Ini membuat pengelolaan *route* menjadi lebih mudah karena kita tidak perlu mengulangi pengaturan yang sama untuk setiap route.

a. Name

Grouping berdasarkan nama memungkinkan kita menggunakan satu nama di dalam route yang sudah kita kelompokkan, sehingga kita tidak perlu menyetikan nama *route* yang sama. Contoh pengelompokan nama seperti ini :

```
// routes/web.php

Route::name('admin.')->group(function(){

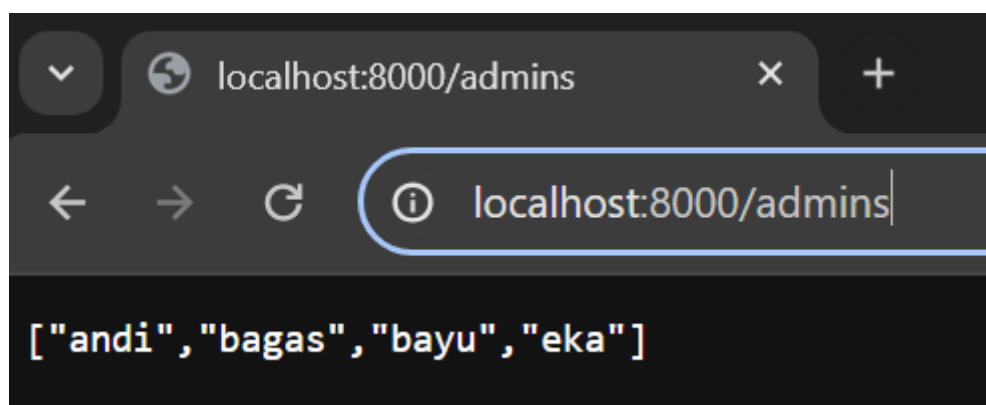
    Route::get("admins/", function(){
        $users = ["andi","bagas","bayu","eka"];
        return $users;
    }->name("list"));

    Route::get("admins/{id}", function($id){
        $users = ["andi","bagas","bayu","eka"];
        return $users[$id];
    }->name("detail"));

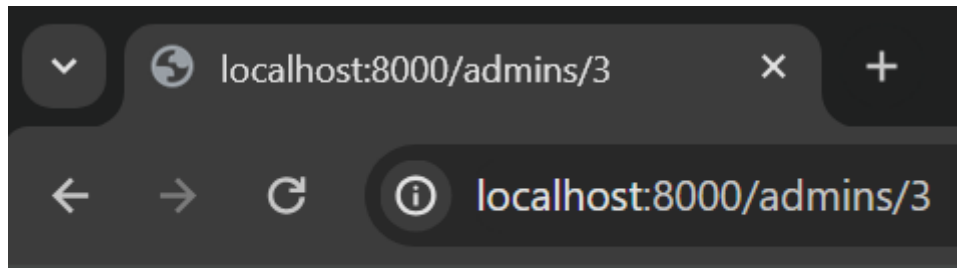
});
```

Contoh Pembuatan Group Berdasarkan Nama.

Ketika kita jalankan di browser kita jalankan kedua route maka akan tampil seperti ini.



Tampilan Dari *Route Admin List*.



eka

Tampilan Dari *Route Admin detail*.

Meskipun sudah dikelompokkan, ketika kita jalankan perintah `artisan route:list`, maka kedua route yang sudah diberi nama akan tetap muncul sama seperti yang sebelumnya.

```
GET|HEAD / ..... homepage
POST _ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionCo...
GET|HEAD _ignition/health-check .... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST _ignition/update-config . ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD admins ..... admin.list
GET|HEAD admins/{id} ..... admin.detail
GET|HEAD beranda .....
GET|HEAD up .....
GET|HEAD users .....
GET|HEAD users/{id?} .....
Showing [10] routes
```

Tampilan Route Yang Diberi Group name.

b. Prefix

Sebelumnya kita sudah mencoba grouping dengan menggunakan *name*, kita juga dapat melakukan pengelompokan URL pada *route*, yaitu dengan menggunakan prefix. Dengan prefix kita dapat mengelompokkan route yang memiliki path URL yang sama, contohnya seperti ini.

```
// routes/web.php

Route::prefix('admins')->group(function(){

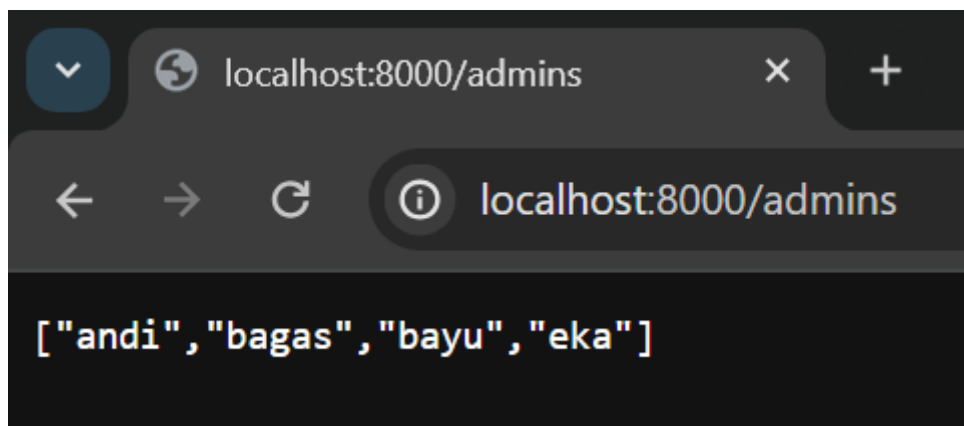
    Route::get("/", function(){
        $users = ["andi","bagas","bayu","eka"];
        return $users;
    });
    // localhost:8000/admins/

    Route::get("/{id}", function($id){
        $users = ["andi","bagas","bayu","eka"];
        return $users[$id];
    });
    // localhost:8000/admins/{id}

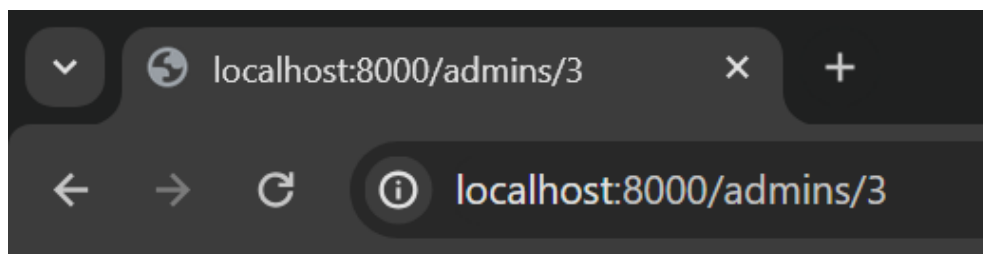
});
```

Contoh Penggunaan Prefix.

Ketika kita akses kedua *route* yang sudah kita kelompokkan dengan prefix di browser maka akan terlihat seperti ini.



Tampilan Dari *Route Admins*.



eka

Tampilan Dari *Route Admins* Dengan Path Yang Dinamis.

Kita juga dapat menggabungkan prefix dan name dalam pengelompokkan, seperti ini.

```
// routes/web.php

Route::prefix('admins')->name("admins.")->group(function(){

    Route::get("/", function(){
        $users = ["andi","bagas","bayu","eka"];
        return $users;
    })->name("list");
    // url: localhost:8000/admins/,
    // name : admins.list

    Route::get("/{id}", function($id){
        $users = ["andi","bagas","bayu","eka"];
        return $users[$id];
    })->name("details");
    // url: localhost:8000/admins/{id},
    // name : admins.list

});
```

Penambahan Name Pada Pengelompokan.

Dengan begitu kita sudah mengelompokkan route sehingga kita tidak lagi mendefinisikan nama dan url yang sama secara berulang. Ketika kita jalankan perintah artisan route:list maka akan terlihat sama seperti sebelumnya.

```
GET|HEAD / ..... homepage
POST _ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionCo...
GET|HEAD _ignition/health-check .... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST _ignition/update-config . ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD admins ..... admins.list
GET|HEAD admins/{id} ..... admins.details
GET|HEAD beranda .....
GET|HEAD up .....
GET|HEAD users .....
GET|HEAD users/{id?} .....

Showing [10] routes
```

Tampilan Route Yang Diberi Group name.

4. Referensi

- <https://laravel.com/docs/11.x/routing>
- <https://laracoding.com/how-to-use-get-parameters-in-urls-using-laravel/>