

LARAVEL SESSION, MIDDLEWARE & AUTHENTICATION

KELAS PROGRAMMING FULL STACK DEVELOPER

MITRA PELATIHAN



Jabar Digital Academy

digitalacademy.jabarprov.go.id

2024

BAB VI

LARAVEL SESSION, MIDDLEWARE, AND AUTHENTICATION.

1. Tujuan

- a. Peserta didik dapat mengetahui apa itu Session, Middleware, dan Authentication.
- b. Peserta didik dapat mengetahui penggunaan Session, Middleware.
- c. Peserta didik dapat mempraktekkan pembuatan authentication dengan memanfaatkan session.

2. Perlengkapan

- a. Modul VI. LARAVEL SESSION, MIDDLEWARE, AND AUTHENTICATION.
- b. IDE atau Teks Editor (Visual Studio Code, Notepad++, Sublime)
- c. Xampp.

3. Materi.

Setiap aplikasi pasti memiliki berbagai macam keamanan yang terpasang di dalamnya, salah satu cara agar aplikasi yang kita kembangkan memiliki keamanan yaitu dengan memberikan authentication. Dengan otentikasi, kita dapat memastikan hanya pengguna yang sah yang memiliki hak akses ke informasi pribadi, transaksi keuangan, atau fitur aplikasi yang sensitif.

Selain itu, kita bisa menggunakan session sebagai penyimpanan data sementara. Dengan memanfaatkan session kita bisa lebih efisien dalam menyimpan data sementara sehingga kita tidak ketergantungan dengan database sebagai penyimpanan data utama.

Middleware berperan penting dalam menyediakan lapisan keamanan, otentikasi, logging, dan berbagai fungsi lainnya dalam aplikasi Laravel. Sehingga dengan middleware, aplikasi kita akan jauh lebih aman dari sisi keamanan.

1) Session.

Session merupakan data yang disimpan sementara. Data session disimpan pada server, sehingga tidak bisa diakses melalui client. Salah satu contoh penggunaan session yang sudah umum adalah untuk membuat proses login dan pengecekan login. Pada proses login, biasanya session menjadi kunci penanda bahwa yang memiliki kunci tersebut bisa masuk ke sistem yang dibuat. Selain session digunakan untuk membuat proses login, session sering juga digunakan untuk menyimpan flash message.

File konfigurasi session aplikasi Anda akan disimpan di `config/session.php`. Secara default, Laravel dikonfigurasi untuk menggunakan file driver session. Opsi konfigurasi session driver menentukan dimana data session akan disimpan untuk setiap request. Berikut beberapa session driver yang tersedia.

- **file** — session disimpan pada lokasi `storage/framework/sessions`.
- **cookie** — session disimpan dalam cookie yang aman dan terenkripsi
- **database** — session disimpan pada relational database.
- **memcached / redis** — session disimpan pada server memcached/redis yang berbasis cache.
- **array** — session disimpan pada array PHP.

Session terdiri key dan value, untuk membuat session kita bisa menggunakan fungsi session yang didalamnya berupa array associative berupa key dan value, contohnya seperti ini.

```
// routes/web.php

Route::get("/setSession", function(){
    session([
        "nama" => "Andre",
        "umur" => 19,
        "asalkota" => "Bandung"
    ]);

    return "Session Telah Ditambahkan";
});
```

Contoh Membuat Session.

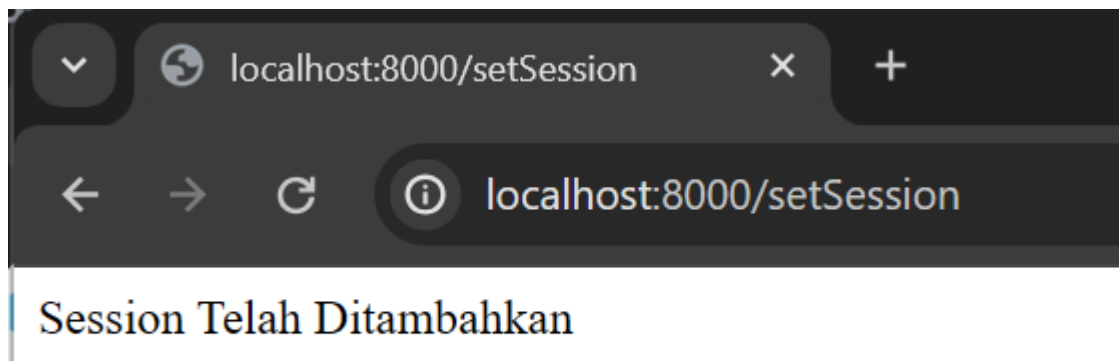
Untuk mendapatkan data dari session, kita bisa menggunakan get lalu tentukan key didalamnya.

```
// routes/web.php

Route::get("/getSession", function(){
    return [
        "nama" => session()->get("nama"),
        "umur" => session()->get("umur"),
        "asalkota" => session()->get("asalkota"),
    ];
});
```

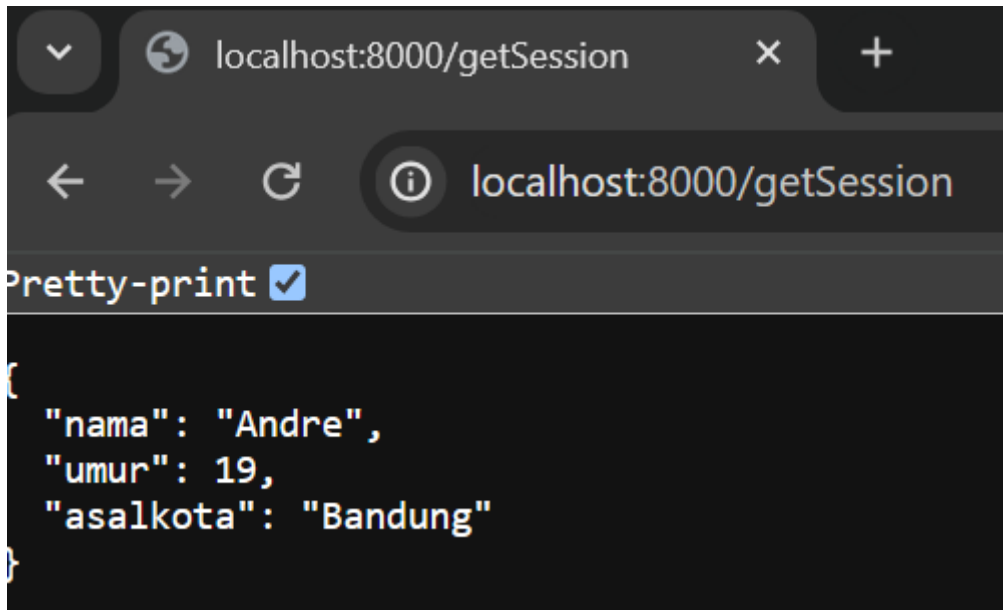
Mendapatkan Data Yang Didapatkan Dari Session.

untuk membuat session, kita bisa akses terlebih dahulu route setSession. Pada saat akses route ini secara otomatis session akan ditambahkan dengan dan akan menampilkan pesan Session telah ditambahkan.



Tampilan Ketika Menjalankan Route SetSession Di Browser.

Lalu ketika kita jalankan route getSession maka data dari session bisa kita dapatkan.



Tampilan Data Yang Didapatkan Menggunakan Get.

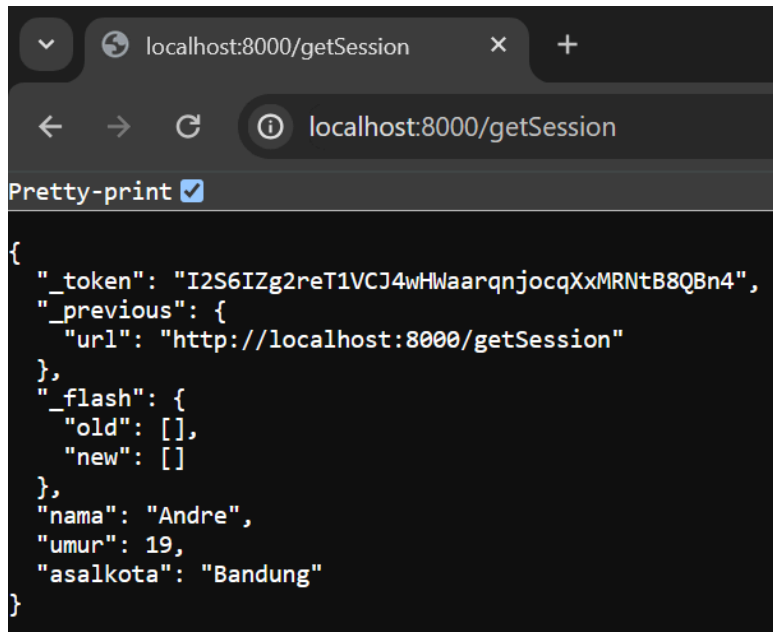
Kita juga bisa mendapatkan semua data yang ada didalam session dengan menggunakan function `all`.

```
// routes/web.php

Route::get("/getSession", function(){
    return session()->all();
});
```

Mendapatkan Semua Data Dari Session.

Ketika kita jalankan di dalam browser, maka semua data yang ada di session akan terlihat seperti ini.



Tampilan Ketika Kita Bisa Menggunakan.

Untuk menghapus session kita bisa menggunakan forget, kita hapus session dengan key nama.

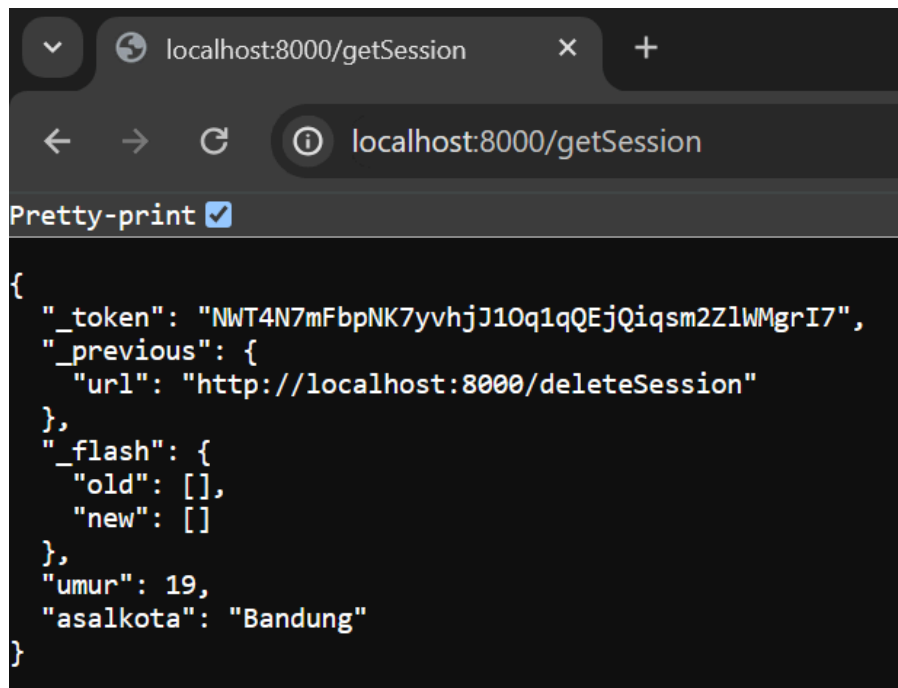
```
// routes/web.php

Route::get("/deleteSession", function(){
    session()->forget("nama");

    return "Session Sudah Dihapus";
});
```

Menghapus Data Session Dengan Forget.

Ketika kita jalankan maka data dari session yang di delete akan null.



Session Nama Dihapus.

Kita juga bisa delete semua session yang dengan fungsi `flush`.

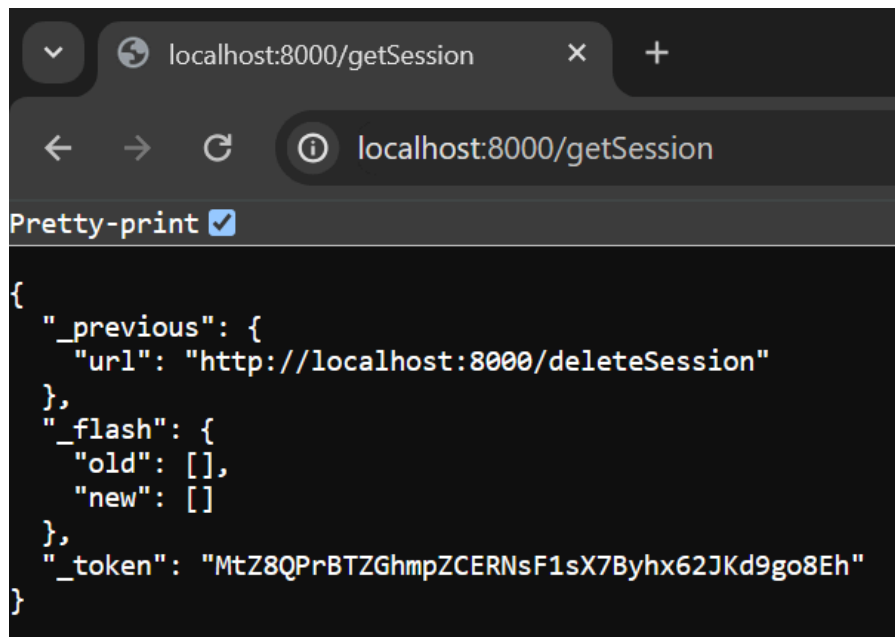
```
// routes/web.php

Route::get("/deleteSession", function(){
    session()->flush();

    return "Session Sudah Dihapus";
});
```

Menghapus Semua Data Session Dengan Flush.

Ketika kita jalankan semua session yang ada akan dihapus.



Tampilan Setelah Semua Data Session Telah Dihapus.

2) Authentication With Session.

Kita akan mencoba membuat authentication dengan menggunakan session, controller, route, dan model yang kita buat di modul lima akan digunakan kembali di dalam contoh kasus ini. Pertama kita buat folder baru bernama `auth` untuk kebutuhan view authentication. Lalu kita buat halaman login beserta form didalamnya.


```

<!-- resources/views/auth/login.blade.php -->

@extends("app")

@section("title", "Halaman Login")
@section("content")

<form action="" method="">
    <h1>Login</h1>

    <label for="email">Email</label>
    <input type="text" id="email" name="email">

    <label for="password">Password</label>
    <input type="password" id="password" name="password">

    <button type="submit">Submit</button>
</form>

@endsection

```

View Form Login.

Lalu kita buat terlebih dahulu controller untuk Authentication, lalu didalamnya kita tambahkan beberapa method yang kita perlukan untuk login.

```

// App/Http/Controllers/AuthController.php;

use Illuminate\Http\Request;

class AuthController extends Controller{
    public function loginPage(){

    }

    public function login(Request $request){

    }

    public function logout(Request $request){

    }
}

```

Method Yang Ada Di Dalam Auth Controller.

Lalu kita buat route untuk halaman login, proses login, dan logout.

```
// routes/web.php

Route::controller(AuthController::class)->prefix("auth")->name("auth.")->group(function(){
    Route::get("/login", "loginPage")->name("loginPage");
    Route::post("/login", "login")->name("login");
    Route::get("/logout", "logout")->name("logout");
});
```

Menambahkan Route Untuk Authenticate.

Setelah kita menambahkan route untuk authentication, kita bisa tambahkan action dan method di dalam form yang ada di dalam halaman login, dan juga kita tambahkan csrf didalamnya.

```
<!-- resources/views/auth/login.blade.php -->

@extends("app")

@section("title", "Halaman Login")
@section("content")

<form action="{{ route("auth.login") }}" method="POST">
    @csrf
    <h1>Login</h1>

    <label for="email">Email</label>
    <input type="text" id="email" name="email">

    <label for="password">Password</label>
    <input type="password" id="password" name="password">

    <button type="submit">Submit</button>
</form>

@endsection
```

kita tambahkan juga return yang mengarah ke view halaman login di dalam method loginPage.

```
// app/Http/Controllers/AuthController.php

use Illuminate\Http\Request;

class AuthController extends Controller{
    public function loginPage(){
        return view('auth.login');
    }
}
```

Menambahkan View Di Dalam Method LoginPage.

Lalu didalam method loginProcess, kita buat alur untuk login yaitu dengan memasukkan email dan password, lalu kita ambil data user berdasarkan email, jika tidak ditemukan maka kembalikan ke halaman login.

```
// App/Http/Controllers/AuthController.blade.php

use App\Models\User;
use Illuminate\Http\Request;

class AuthController extends Controller{
    ...

    public function login(Request $request){
        $user = User::where("email", $request->email)->first();

        if($user == null){
            return redirect()->back();
        }
    }
}
```

Melakukan Check Pada User Yang Emailnya Sesuai.

Kita juga bisa menambahkan pesan error ketika kita yang disimpan didalam session, yaitu dengan menggunakan fungsi with.

```
// App/Http/Controllers/AuthController.blade.php

use App\Models\User;
use Illuminate\Http\Request;

class AuthController extends Controller{
    ...

    public function login(Request $request){
        $user = User::where("email", $request->email)->first();

        if($user == null){
            return redirect()->back()->with("error", "User Tidak Ditemukan!!");
        }
    }
}
```

Menambahkan With Untuk Menyimpan Pesan Error Ke Dalam Session.

Lalu didalam view login kita tambahkan pesan error yang akan terlihat ketika login gagal.

```
<!-- resources/views/auth/login.blade.php -->

@extends("app")

@section("title", "Halaman Login")
@section("content")

@if(session()->get("error"))
    <p style="color: red"> {{ session()->get("error")} }</p>
@endif
<form action="{{ route("login.process") }}" method="POST">
    @csrf
    <h1>Login</h1>

    <label for="email">Email</label>
    <input type="text" id="email" name="email">

    <label for="password">Password</label>
    <input type="password" id="password" name="password">

    <button type="submit">Submit</button>
</form>

@endsection
```

Menampilkan Pesan Error Ketika Gagal Melakukan Login.

Selain kita mendapatkan data berdasarkan email yang sesuai dengan yang dikirim dari halaman login. Kita juga perlu melakukan check pada password yang dikirim dari halaman login untuk disesuaikan dengan password yang ada didalam database. untuk melakukan check pada password kita bisa menggunakan fungsi `check` yang ada di dalam class `hash`.

```
// App/Http/Controllers/AuthController.blade.php

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class AuthController extends Controller{
    ...

    public function login(Request $request){
        $user = User::where("email", $request->email)->first();

        if($user == null){
            return redirect()->back()->with("error", "User Tidak Ditemukan!!");
        }

        if(!Hash::check($request->password, $user->password)){
            return redirect()->back()->with("error", "Password Salah!!");
        }
    }
}
```

Penambahan Check Pada Password.

Jika password yang diberikan tidak sesuai dengan password yang disimpan didalam database, maka kita arahkan kembali ke halaman login dan berikan pesan error. Namun jika password sudah sesuai maka kita buat data ke dalam session untuk mengidentifikasi bahwa dia sudah login. Dan yang login role nya akan menjadi user.

```
// app/Http/Controllers/AuthController.php

class AuthController extends Controller{
    ...

    public function login(Request $request){
        $user = User::where("email", $request->email)->first();

        if($user == null){
            return redirect()->back()->with("error", "User Tidak Ditemukan!!");
        }

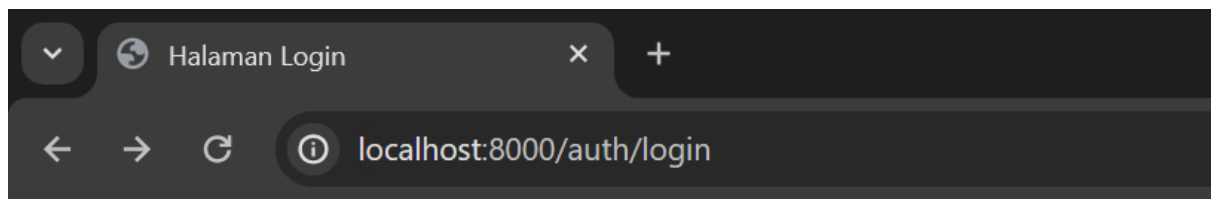
        if(!Hash::check($request->password, $user->password)){
            return redirect()->back()->with("error", "Password Salah!!");
        }

        $request->session()->regenerate();
        $request->session()->put('isLoggedIn', true);
        $request->session()->put("userId", $user->id);
        $request->session()->put("role", "user");

        return redirect()->route("users.index");
    }
}
```

Menambahkan Data Ke Dalam Session.

Ketika kita jalankan maka akan terlihat seperti ini.

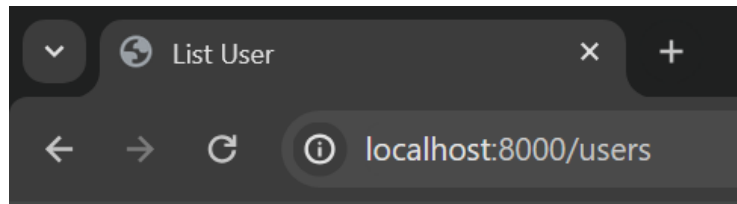


Login

Email Password

Tampilan Halaman Login.

Ketika kita submit, maka akan diarahkan ke halaman index users.

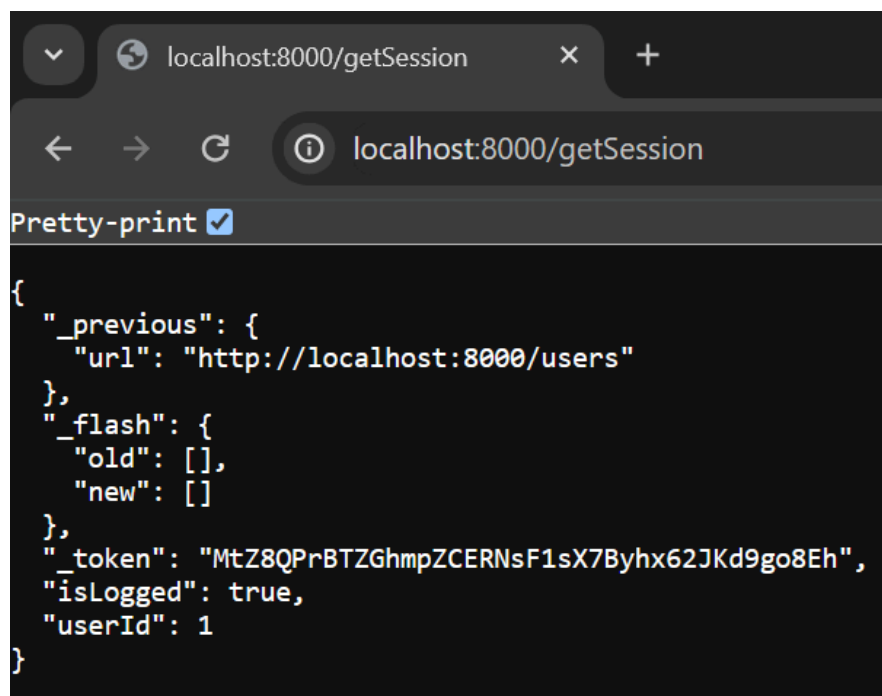


[Tambah User](#)

No	Nama	Jumlah Task	aksi	
1	Mylene Mills	5	Edit	Hapus
2	Riko nikko	0	Edit	Hapus

Tampilan Ketika Diarahkan Ke Halaman Index User.

Untuk check apakah user itu sudah login atau belum bisa dilihat di session.



Data Di Dalam Session Ketika User Sudah Login.

Untuk melakukan logout, kita hanya perlu menghapus data yang ada di dalam session.

```
// app/Http/Controllers/AuthController.php

class AuthController extends Controller{
    ...

    public function logout(Request $request){
        session()->forget("isLoggedIn");
        session()->forget("userId");
        session()->forget("role");

        return redirect()->route("auth.login");
    }
}
```

Menghapus Session Menggunakan Forget.

Kita juga bisa menggunakan `flush` untuk menghapus semua data yang ada di dalam session.

```
// app/Http/Controllers/AuthController.php

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class AuthController extends Controller{
    ...

    public function logout(Request $request){
        session()->flush();

        return redirect()->route("auth.login");
    }
}
```

Menghapus Session Menggunakan Flush.

Kita tambahkan logout di dalam view index user.


```

<!-- resources/views/users/index.blade.php -->

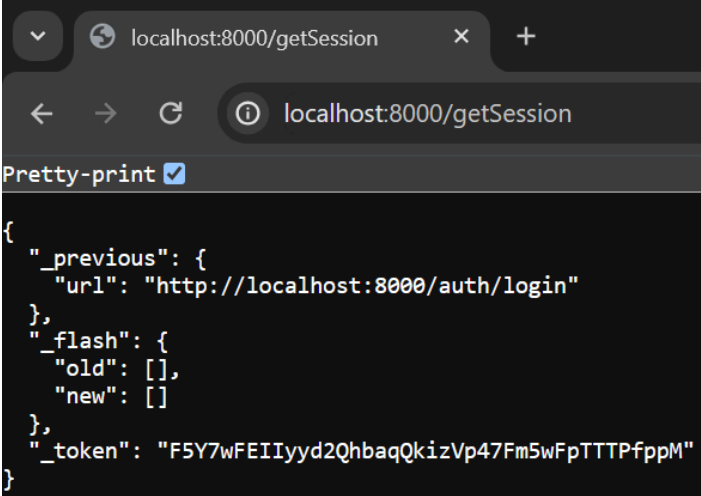
@extends("app")

@section("title","List User")
@section("content")
    @if(session()->get("isLoggedIn"))
        <a href="{{ route("auth.logout") }}">Logout</a>
    @endif
    <a href="{{ route("users.create") }}">Tambah User</a>
    <table border="1">
        <thead>
            <tr>
                <th>No</th>
                <th>Nama</th>
                <th>Jumlah Task</th>
                <th colspan="2">aksi</th>
            </tr>
        </thead>
        <tbody>
            @foreach($users as $index => $user)
                <tr>
                    <td>{{ $index+1 }}</td>
                    <td><a href="{{ route("users.show", $user->id) }}">{{ $user->name }}</a></td>
                    <td>{{ count($user->task) }}</td>
                    <td><a href="{{ route("users.edit", $user->id) }}">Edit</a></td>
                    <td><form action="{{ route("users.destroy", $user->id) }}" method="POST">
                        @csrf
                        @method("DELETE")
                        <button type="submit">Hapus</button>
                    </form></td>
                </tr>
            @endforeach
        </tbody>
    </table>
@endsection

```

Menambahkan Logout Di Dalam View Index User.

Ketika kita klik logout di halaman index users maka browser akan diarahkan ke halaman login, maka data yang ada di dalam session akan dihapus.



```

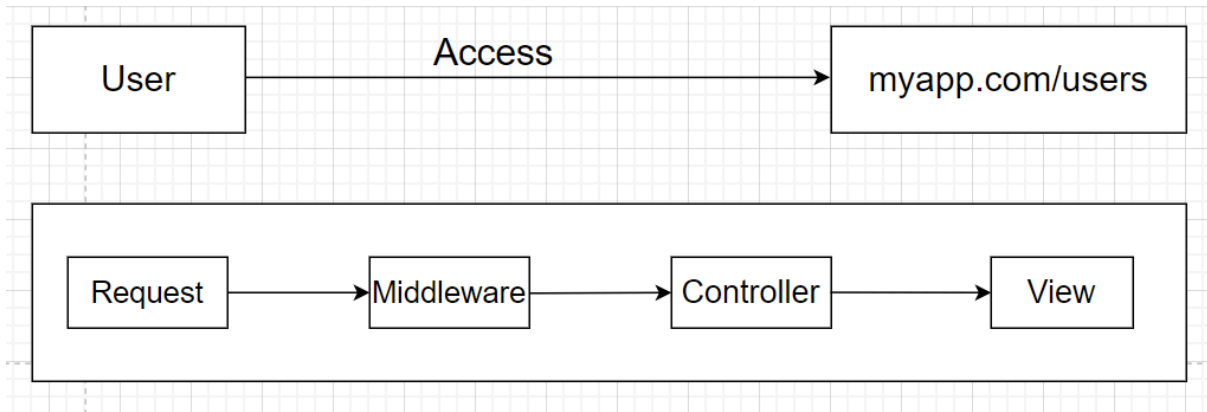
{
  "_previous": {
    "url": "http://localhost:8000/auth/login"
  },
  "_flash": {
    "old": [],
    "new": []
  },
  "_token": "F5Y7wFEIIyyd2QhbaqQkizVp47Fm5wFpTTTPfppM"
}

```

Data Login Yang Ada Di Session Setelah Dihapus.

3) Middleware.

Middleware menyediakan mekanisme untuk memeriksa dan memfilter permintaan HTTP yang masuk ke aplikasi Anda. Middleware dijalankan sebelum Controller dijalankan.



Alur Ketika System Menjalankan Aplikasi Kita.

Misalnya, kita menyertakan middleware yang memverifikasi bahwa pengguna aplikasi Anda diautentikasi. Semua middleware ini terletak di direktori `app/Http/Middleware`, namun ketika kita pertama kali install laravel, folder ini secara default tidak ada, folder ini akan ditambahkan ketika membuat middleware dengan perintah artisan.

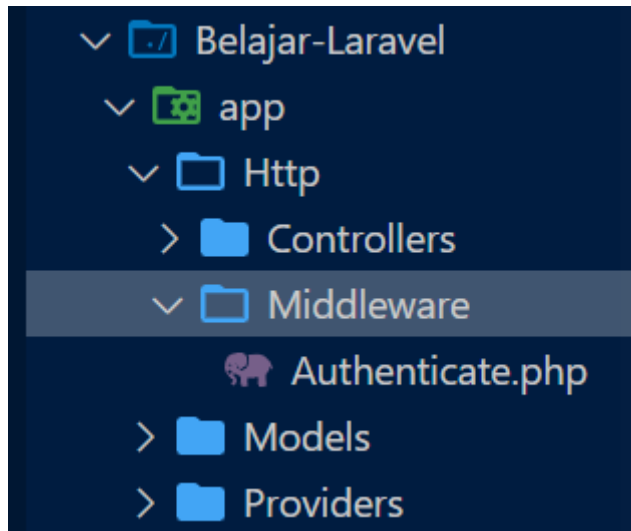
`php artisan make:middleware (nama_middleware)`

Contoh middleware yang akan kita buat gunakan untuk membatasi user yang sedang tidak login untuk tidak bisa mengakses suatu halaman. Untuk membuat middleware menggunakan perintah artisan seperti ini.

```
advan@DESKTOP-K34IR2I MINGW64 /d/Programing/Belajar-Laravel
○ $ php artisan make:middleware Authenticate
```

Pembuatan Middleware Dengan Perintah Artisan.

Jika sudah dibuat kita bisa lihat middleware yang sudah dibuat didalam folder `app/Http/Middleware`.



Tampilan Folder Middleware.

Secara default, middleware akan secara otomatis dibuatkan fungsi handle. Ketika middleware dijalankan, maka yang akan semua code yang ada di dalam fungsi handle yang akan dijalankan. Jadi seluruh logic atau code yang perlu dijalankan terlebih dahulu sebelum controller dijalankan bisa kita isi disini.

```
// app/Http/middleware/authenticate.php

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class Authenticate{
    public function handle(Request $request, Closure $next): Response{
        return $next($request);
    }
}
```

Middleware Yang Sudah Dibuat.

Karena kita ingin membuat middleware untuk melakukan check apakah aplikasi ini dibuka dalam keadaan user sedang login atau tidak, maka kita bisa tambahkan kondisi yang di dalamnya adalah mengambil data, jika kondisi terpenuhi maka akan diarahkan ke halaman login. Seperti ini.

```
// app/Http/middleware/authenticate.php

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class Authenticate{
    public function handle(Request $request, Closure $next): Response{
        if(session()->get("isLoggedIn") == null && session()->get("userId") == null){
            return redirect()->route("auth.login");
        }

        return $next($request);
    }
}
```

Penambahan Kondisi Baru Untuk Memeriksa User Login.

Kita juga bisa menambahkan pesan error pada saat diarahkan ke halaman login dengan menggunakan with.

```
// app/Http/middleware/authenticate.php

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class Authenticate{
    public function handle(Request $request, Closure $next): Response{
        if(session()->get("isLoggedIn") == null && session()->get("userId") == null){
            return redirect()->route("auth.login")->with("error", "Perlu Login Terlebih Dahulu!!");
        }

        return $next($request);
    }
}
```

Menambahkan With Untuk Menyimpan Pesan Error Ke Dalam Session.

Untuk memasang middleware ke dalam halaman yang ingin kita batasi aksesnya ketika sedang login, kita bisa tambahkan fungsi middleware didalam route yang akan diberi middleware, seperti ini.

```

use App\Http\Middleware\Authenticate;

Route::controller(UserController::class)->prefix("users")->name("users.")->group(function(){
    Route::get("/", "index")->name("index")->middleware(Authenticate::class);
    Route::get("/create", "create")->name("create")->middleware(Authenticate::class);
    Route::get("/{id}", "show")->name("show")->middleware(Authenticate::class);
    Route::get("/{id}/edit", "edit")->name("edit")->middleware(Authenticate::class);

    Route::post("/", "store")->name("store")->middleware(Authenticate::class);
    Route::put("/{id}", "update")->name("update")->middleware(Authenticate::class);

    Route::delete("/{id}", "destroy")->name("destroy")->middleware(Authenticate::class);
});

```

Menambahkan Middleware Ke Dalam Route User.

Kita juga bisa melakukan grouping pada middleware sehingga kita tidak perlu mendefinisikan satu persatu middleware yang akan dipasang, seperti ini.

```

// routes/web.php

use App\Http\Middleware\Authenticate;
use App\Http\Controllers\UserController;

Route::controller(UserController::class)->middleware(Authenticate::class)->prefix("users")->name("users.")->group(function(){
    Route::get("/", "index")->name("index");
    Route::get("/create", "create")->name("create");
    Route::get("/{id}", "show")->name("show");
    Route::get("/{id}/edit", "edit")->name("edit");

    Route::post("/", "store")->name("store");
    Route::put("/{id}", "update")->name("update");

    Route::delete("/{id}", "destroy")->name("destroy");
});

```

Menambahkan Grouping Pada Middleware.

Ketika kita coba akses route yang sudah kita pasang middleware dalam keadaan tidak login, maka kita akan secara otomatis diarahkan ke halaman login. Kita juga bisa memasang lebih dari satu middleware, yaitu dengan menyisipkan array yang di dalamnya berupa middleware yang ingin dipasang, seperti ini.

```
// routes/web.php

use App\Http\Middleware\Authenticate;
use App\Http\Controllers\UserController;

Route::controller(UserController::class)->middleware([Authenticate::class])->prefix("users")->name("users.")->group(function(){
    Route::get("/", "index")->name("index");
    Route::get("/create", "create")->name("create");
    Route::get("/{id}", "show")->name("show");
    Route::get("/{id}/edit", "edit")->name("edit");

    Route::post("/", "store")->name("store");
    Route::put("/{id}", "update")->name("update");

    Route::delete("/{id}", "destroy")->name("destroy");
});
```

Menyisipkan Array Di Dalam Middleware.

Kalau kita ingin menjalankan middleware disetiap route yang ada. Kita bisa atur middleware mana saja yang ingin kita jalankan di semua route di dalam sebuah file yang ada di dalam `bootstrap/app.php`. File `app` terdapat tiga fungsi yang digunakan untuk konfigurasi aplikasi kita. Pertama adalah `withRouting` digunakan untuk menentukan dimana kita akan menuliskan routing, secara default kita menuliskan route di dalam folder `route/web.php`.

```
// bootstrap/app.php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__ . '/../routes/web.php',
        commands: __DIR__ . '/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function(Middleware $middleware){ // used for global middleware.
        //
    })
    ->withExceptions(function(Exceptions $exceptions){
        //
    })->create();
```

Konfigurasi Yang Ada Di Dalam File `bootstrap/app.php`.

Lalu setelahnya ada `withMiddleware`, yang kita gunakan untuk mengatur middleware yang kita buat, contohnya seperti kita bisa jalankan middleware di semua routing yang kita punya, yaitu dengan menggunakan `append`. Contohnya seperti ini.

```
// bootstrap/app.php

use App\Http\Middleware\Authenticate;
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function(Middleware $middleware){ // used for global middleware.
        $middleware->append(Authenticate::class);
    })
    ->withExceptions(function(Exceptions $exceptions){
        //
    })->create();
```

Menambahkan Append Didalam WithMiddleware.

Jika kita ingin menjalankan banyak middleware di semua routing yang kita miliki, kita bisa menggunakan fungsi `use` yang di dalamnya diisi dengan array yang berisikan middleware mana saja yang akan dijalankan.

```
use App\Http\Middleware\Authenticate;
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function(Middleware $middleware){ // used for global middleware.
        $middleware->use([
            Authenticate::class
        ]);
    })
    ->withExceptions(function(Exceptions $exceptions){
        //
    })->create();
```

Menambahkan Append Didalam WithMiddleware.

Kita juga bisa memberikan alias pada middleware yang kita buat, yaitu dengan menggunakan fungsi `alias`.

```

use App\Http\Middleware\Authenticate;
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__ . '/../routes/web.php',
        commands: __DIR__ . '/../routes/console.php',
        health: 'up',
    )
    ->withMiddleware(function(Middleware $middleware){ // used for global middleware.
        $middleware->alias([
            "authenticate" => Authenticate::class
        ]);
    })
    ->withExceptions(function(Exceptions $exceptions){
        //
    })->create();

```

Menambahkan Alias Di Dalam WithMiddleware.

dengan memberikan alias pada middleware kita, kita tidak perlu melakukan pemanggilan kelas middleware secara langsung pada saat ingin menggunakan middleware didalam route, cukup kita beri nilai string dari nilai alias yang kita berikan. Seperti ini.

```

// routes/web.php

Route::controller(UserController::class)->middleware(["authenticate"])->prefix("users")->name("users.")->group(function(){
    Route::get("/", "index")->name("index");
    Route::get("/create", "create")->name("create");
    Route::get("/{id}", "show")->name("show");
    Route::get("/{id}/edit", "edit")->name("edit");

    Route::post("/", "store")->name("store");
    Route::put("/{id}", "update")->name("update");

    Route::delete("/{id}", "destroy")->name("destroy");
});

```

Memanggil Middleware Dengan Alias.

Selain itu kita juga bisa menyisipkan data ke dalam argumen pada middleware yang telah kita buat pada saat kita ingin memanggil fungsi tersebut, contohnya kita coba tambahkan parameter baru di dalam middleware yang sudah kita buat.


```
// app/Http/Middleware/Authenticate.php

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class Authenticate{
    public function handle(Request $request, Closure $next, string $role): Response{
        if(
            session()->get("isLoggedIn") == null &&
            session()->get("userId") == null
        ){
            return redirect()->route("auth.login")->with("error", "Perlu Login Terlebih Dahulu !!");
        }

        if(session()->get("role") != $role){
            return redirect()->route("auth.login")->with("error", "Anda Tidak Memiliki Akses!!");
        }

        return $next($request);
    }
}
```

Menambahkan Parameter Baru Dan Kondisi.

Untuk memasukkan nilai ke dalam argumen dalam middleware, kita cukup menambahkan nilai setelah titik dua di dalam definisi middleware tersebut. Namun, jika kita ingin menyisipkan data ke dalam argumen, kita perlu membuat alias terlebih dahulu pada middleware yang akan kita gunakan. Sebagai contoh:

```
// app/Http/Middleware/Authenticate.php

Route::controller(UserController::class)->middleware(["authenticate:user"])->prefix("users")->name("users.")->group(function(){
    Route::get("/", "index")->name("index");
    Route::get("/create", "create")->name("create");
    Route::get("/{id}", "show")->name("show");
    Route::get("/{id}/edit", "edit")->name("edit");

    Route::post("/", "store")->name("store");
    Route::put("/{id}", "update")->name("update");

    Route::delete("/{id}", "destroy")->name("destroy");
});
```

Memasukkan Nilai Ke Dalam Argumen Pada Class Middleware.

4. Referensi.

- <https://laravel.com/docs/11.x/middleware#middleware-parameters>
- <https://laravel.com/docs/11.x/session#main-content>