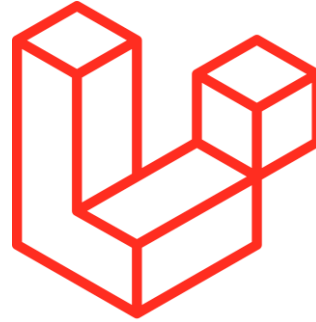


Pemrograman Backend



Pesantren PeTIK II YBM PLN

Jl. KH. Bisri Syansuri RT/01 RW/05, Plosogeneng,
Kec. Jombang, Kabupaten Jombang, Jawa Timur



12. Session dan Otentikasi User



Definisi Session(1)

Session adalah mekanisme untuk mempertahankan informasi di semua halaman web yang berbeda untuk mengidentifikasi pengguna saat mereka menelusuri situs atau aplikasi. Apakah Anda bertanya-tanya mengapa session dibutuhkan untuk sebuah aplikasi web? Untuk melihat kenapa session dibutuhkan, kita harus kembali ke belakang dan melihat bagaimana protokol HTTP dirancang untuk bekerja.





Definisi Session(2)

- Protokol HTTP merupakan protokol yang tidak memiliki state, artinya tidak mungkin server dapat mengingat pengguna tertentu di antara beberapa request. Contohnya, ketika Anda mengakses sebuah halaman web, server hanya bertanggung jawab untuk menyediakan konten dari halaman di-request tersebut. Jadi ketika Anda mengakses halaman lain dari website yang sama, server web menginterpretasi setiap dan semua request secara terpisah, seolah-olah mereka tidak berhubungan satu sama lain. Tidak ada kemungkinan bagi server untuk dapat mengetahui setiap request berasal dari pengguna yang sama.
- Session memungkinkan Anda untuk berbagi informasi ke semua halaman yang berbeda dalam satu situs atau aplikasi, sehingga dapat membantu menjaga state. Ini memungkinkan server mengetahui bahwa semua request berasal dari pengguna yang sama, sehingga situs bisa menampilkan informasi spesifik dari pengguna serta preferensi.





Definisi Otentikasi User

- Authentication atau Otentikasi adalah suatu proses atau tindakan untuk membuktikan atau menunjukkan sesuatu yang benar, asli, atau valid.
- Teknologi otentikasi menyediakan kontrol akses untuk sistem dengan memeriksa atau melihat apakah kredensial pengguna cocok dengan kredensial di dalam database pengguna yang berwenang atau di server otentikasi data.
- Pengguna biasanya diidentifikasi dengan ID pengguna, dan otentikasi dilakukan ketika pengguna memberikan kredensial apapun.
- Bentuk kredensial ini misalnya kata sandi yang nantinya akan dicocokkan dengan ID pengguna tersebut.
- Sebagian besar pengguna terbiasa menggunakan kata sandi sebagai bagian dari informasi yang seharusnya hanya diketahui pengguna.
- Pengetahuan pengguna akan kata sandi ini disebut sebagai faktor otentikasi.
- Faktor otentikasi lainnya adalah bagaimana cara mereka digunakan untuk otentikasi dua faktor atau otentikasi multifaktor.





Install Extensions Laravel/ui

Untuk membuat fitur otentikasi user di Laravel, kita bisa menggunakan extension laravel/ui. Untuk menginstallnya masuk ke direktori project, lalu lakukan perintah di bawah ini:

```
C:\xampp\htdocs\laraveltm4>composer require laravel/ui
Using version ^2.0 for laravel/ui
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing laravel/ui (v2.0.3): Downloading (100%)
Writing lock file
Generating optimized autoload files
```



Menu Help Laravel/ui

Setelah selesai proses installasi vendor Laravel/ui, langkah selanjutnya adalah kita bisa melihat menu helpnya dengan perintah sebagai berikut:

```
C:\xampp\htdocs\laraveltm4>php artisan ui --help
Description:
  Swap the front-end scaffolding for the application

Usage:
  ui [options] [--] <type>

Arguments:
  type                The preset type (bootstrap, vue, react)

Options:
  --auth              Install authentication UI scaffolding
  --option[=OPTION]  Pass an option to the preset command (multiple)
  -h, --help          Display this help message
  -q, --quiet         Do not output any message
  -V, --version       Display this application version
  --ansi             Force ANSI output
  --no-ansi          Disable ANSI output
  -n, --no-interaction Do not ask any interactive question
  --env[=ENV]        The environment the command should run under
```



User Interface Otentikasi User

Langkah selanjutnya ada menjalankan perintah untuk instalasi otentikasi user pada Laravel dengan Langkah-langkah sebagai berikut:

- Buka terminal/CMD lalu masuk ke folder project Laravel.
- Setelah itu ketikkan perintah: `php artisan ui bootstrap --auth`.
- Untuk user interfacesnya kita bisa memilih selain bootstrap, yaitu : vue dan react.

```
C:\xampp\htdocs\laraveltm4>php artisan ui bootstrap --auth
Bootstrap scaffolding installed successfully.
Please run "npm install && npm run dev" to compile your fresh scaffolding.
Authentication scaffolding generated successfully.
```



Instalasi Node.js

Untuk menginstall user interface (ui) otentikasi user, pada Laravel dibutuhkan software node.js. Untuk menginstall Node.js pada MS Windows, kita perlu mendownload aplikasi Node.js pada situs web <https://nodejs.org/en/>.



Proses Install Otentikasi User

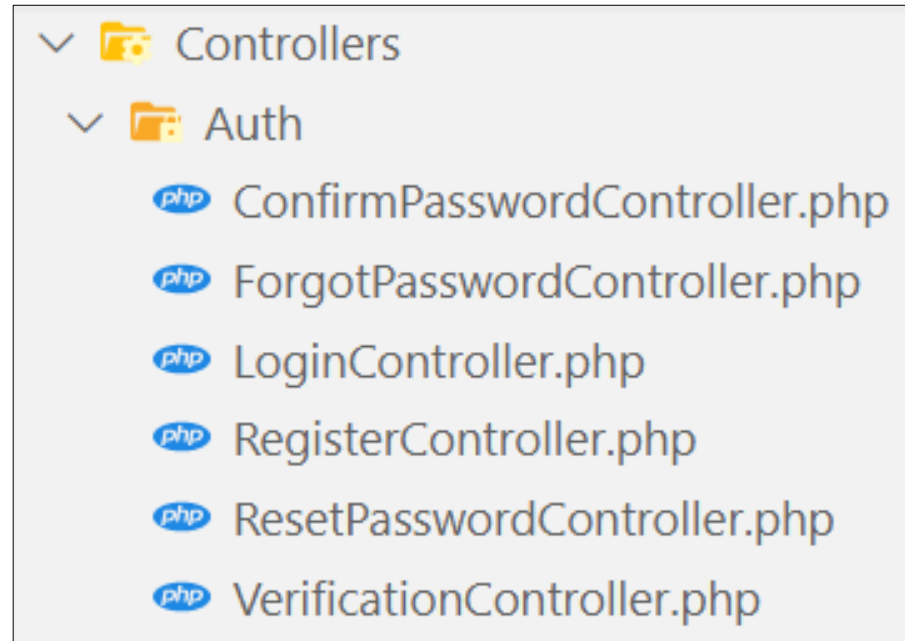
Langkah selanjutnya adalah menginstall dan menjalankan ui otentikasi user melalui software node.js dengan perintah berikut ini:

```
C:\xampp\htdocs\laraveltm4>npm install && npm run dev
npm WARN deprecated popper.js@1.16.1: You can find the
legacy v1
npm WARN deprecated chokidar@2.1.8: Chokidar 2 will bre
npm WARN deprecated fsevents@1.2.12: fsevents 1 will bre
sevents 2.
npm notice created a lockfile as package-lock.json. You
```



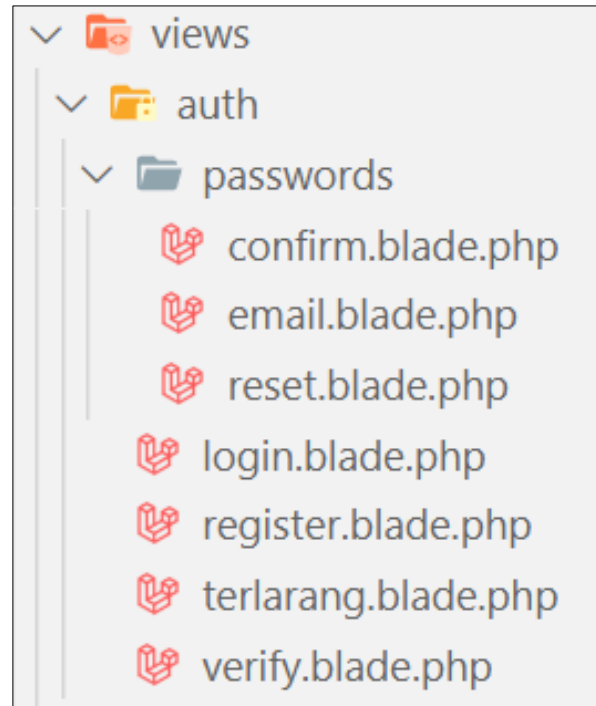
Generate Controllers Auth

Setelah selesai proses instalasi vendor laravel/ui, maka secara otomatis akan dibuatkan controller terkait dengan otentikasi user di folder app/Controllers/Auth dengan file-file sebagai berikut:



Generate Views Auth

Selain dibuatkan controller secara otomatis, maka akan degenerate pula vies terkait dengan otentikasi user di folder resources/views/auth dengan file-file sebagai berikut:



Routing Auth dengan Middleware

Setelah berhasil menggunakan fitur otentikasi user, langkah selanjutnya adalah menambahkan routing di routes/web.php yang bertujuan untuk membatasi hak akses user dengan konsep middleware laravel seperti kode program di dalam bawah ini:

```
Route::resource('pengarang', PengarangController::class)->middleware('auth');
Route::resource('penerbit', PenerbitController::class)->middleware('auth');
Route::resource('kategori', KategoriController::class)->middleware('auth');
Route::resource('buku', BukuController::class)->middleware('auth');
```

```
Auth::routes();
Route::get('/home',
[App\Http\Controllers\HomeController::class, 'index'])->name('home');
```



Migrasi Database(1)

Setelah berhasil membuat user interface otentikasi user pada Laravel, langkah selanjutnya adalah memigrasi database untuk mendapatkan tabel terkait otentikasi user di dalam database secara otomatis dengan perintah sebagai berikut:

```
C:\xampp\htdocs\laraveltm4>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table (0.28 seconds)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table (0.28 seconds)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated: 2019_08_19_000000_create_failed_jobs_table (0.19 seconds)
```





Migrasi Database(2)

Setelah proses migrasi database berhasil, sekarang lihatlah struktur data dalam database Anda, maka ada penambahan tabel-tabel sebagai berikut:

Server: 127.0.0.1 » Database: dbperpus

Structure SQL Search Query Export Import Operations Privileges Routines Events More

Filters

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> anggota	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	64.0 KiB	-
<input type="checkbox"/> buku	★ Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	80.0 KiB	-
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> kategori	★ Browse Structure Search Insert Empty Drop	5	InnoDB	latin1_swedish_ci	16.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> password_resets	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> peminjaman	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	48.0 KiB	-
<input type="checkbox"/> penerbit	★ Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	32.0 KiB	-
<input type="checkbox"/> pengarang	★ Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	16.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
10 tables	Sum	18	InnoDB	utf8mb4_general_ci	352.0 KiB	0 B

Migrasi Database(3)

Pada tabel users kita bisa memodifikasi dengan menambahkan field-field seperti:

- role untuk keperluan otorisasi user yang kita buat sendiri.
- Isactive untuk menandakan user tersebut aktif atau tidak.
- foto untuk upload foto user.

#	Name	Type
1	id 🔑	bigint(20)
2	name	varchar(255)
3	email 🔑	varchar(255)
4	email_verified_at	timestamp
5	password	varchar(255)
6	role	enum('admin', 'staff', 'anggota')
7	isactive	enum('yes', 'no', 'banned')
8	foto	varchar(30)
9	remember_token	varchar(100)
10	created_at	timestamp
11	updated_at	timestamp



Redirect Setelah Login

Untuk mengarahkan user ke sebuah halaman (landing page) ketika berhasil login, Anda cukup mengubah di file LoginController.php yang berada di folder app/Http/Controllers/Auth seperti tampak di dalam slide dengan penjelasan kode program sebagai berikut:

```
use AuthenticatesUsers;
```

```
/**  
 * Where to redirect users after login.  
 *  
 * @var string  
 */
```

```
//protected $redirectTo = RouteServiceProvider::HOME;  
protected $redirectTo = '/buku';
```

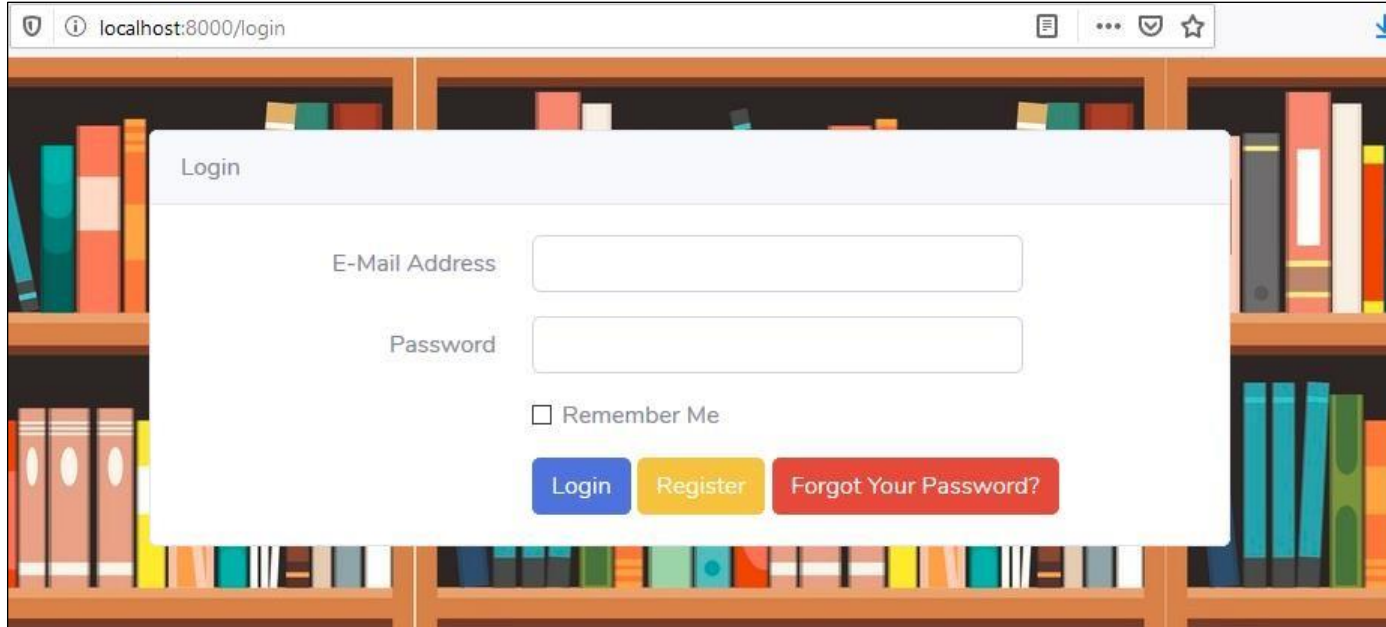
Modifikasi Login

Untuk menambahkan sistem otentikasi user selain verifikasi username dan password, misalnya dengan menambahkan field isactive = yes pada aplikasi Laravel, maka Anda cukup mengubahnya di file AuthenticatesUsers.php pada fungsi credentials() yang berada di folder vendor/Laravel/ui/auth-backend seperti tampak di dalam slide dengan penjelasan kode program sebagai berikut:

```
protected function credentials(Request $request)
{
    //return $request->only($this->username(), 'password');
    return array_merge(
        $request->only($this->username(), 'password'),
        ['isactive' => 'yes']
    );
}
```

View Login

Untuk melihat tampilan login, pada web browser di url ketikkan `http://localhost:8000/login`



localhost:8000/login

Login

E-Mail Address

Password

☐ Remember Me

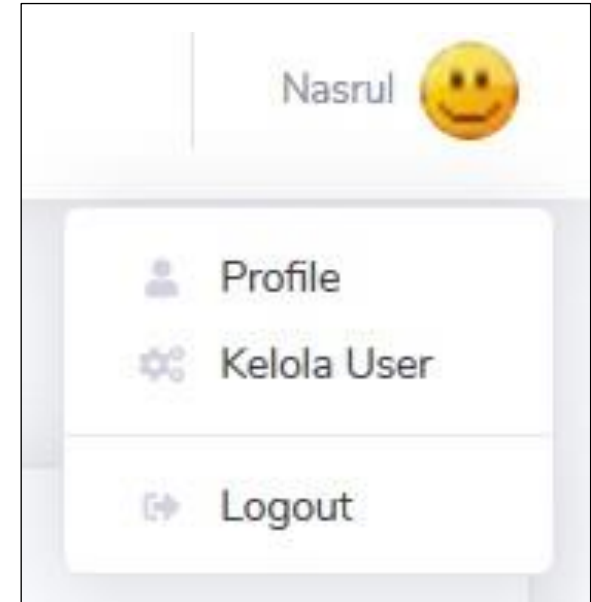
[Login](#) [Register](#) [Forgot Your Password?](#)



Menampilkan User

Setelah Anda berhasil membuat fitur otentikasi user, Langkah selanjutnya mencetak user yang sudah berhasil login pada halaman aplikasi. Kita bisa menambahkannya misal di halaman navbar.blade.php dengan kode program sebagai berikut:

```
<a class="nav-link dropdown-toggle" href="#">
    @if(empty(Auth::user()->name))
    {{ ' ' }}
    @else
    {{ Auth::user()->name }}
    @endif
    
</a>
```



Fitur Logout

Untuk mengakhiri session login, Anda perlu menambahkan fitur logout pada aplikasi Anda. Untuk membuat fitur logout, Anda cukup mengarahkan menu logout ke routing logout bawaan dari vendor seperti tampak di dalam slide dengan penjelasan kode program sebagai berikut:

```
<a class="dropdown-item" href="{{ route('logout') }}"
    onclick="event.preventDefault();
    document.getElementById(
        'logout-form').submit();">
    {{__('Logout')}}
</a>
<form id="logout-form" action="{{ route('logout') }}"
    method="POST" style="display: none;">
    @csrf
</form>
```



Fitur Register User(1)

Setelah Anda berhasil membuat fitur otentikasi user, langkah selanjutnya menambahkan fitur registrasi user yang sudah ada di dalam vendor Laravel/ui. Yang perlu kita ubah pertama kali adalah mengubah landing page halaman setelah user melakukan registrasi, dengan mengubah file Registerusers.php yang ada di folder vendor/Laravel/ui/auth-backend.

```
use RegistersUsers;

/**
 * Where to redirect users after registration.
 *
 * @var string
 */
//protected $redirectTo = RouteServiceProvider::HOME;
protected $redirectTo = '/afterregister';
```

Fitur Register User(2)

Untuk selanjutnya kita buat halaman setelah melakukan registrasi, misal: afterRegister.blade.php seperti kode program di bawah ini:

```
@section('content')
<div class="container">
<div class="jumbotron">
    <h2>
        Terima Kasih sudah Register
    </h2>
    <p>
        Mohon tunggu approval dari Administrator Kami
    </p>
</div>
</div>
@endsection
```

Fitur Register User(3)

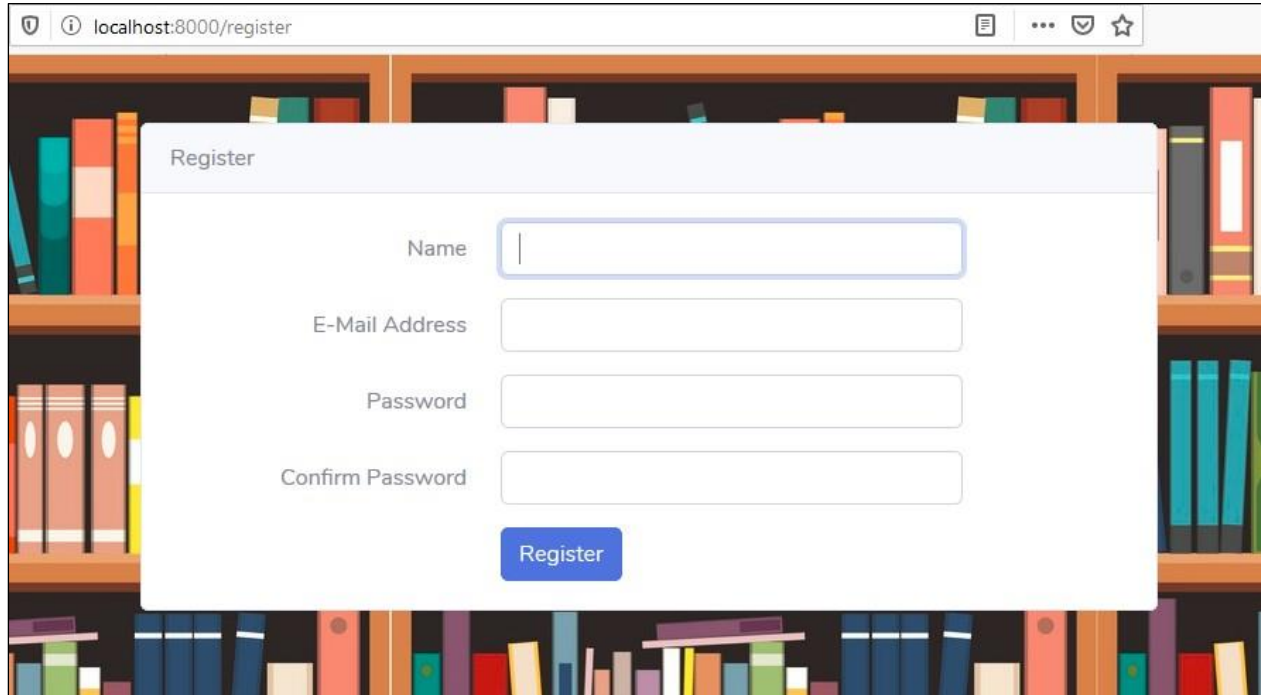
Ketika user berhasil melakukan registrasi secara default Laravel langsung menjadikannya member user. Untuk mengubahnya dengan sistem persetujuan dari admin, maka Anda cukup mengubahnya di file Registerusers.php yang berada di folder vendor/Laravel/ui/auth-backend. Agar tidak langsung otomatis menjadi member user, Anda cukup menonaktifkan kode **`$this->guard()->login($user);`** seperti kode program di bawah ini:

```
public function register(Request $request)
{
    $this->validator($request->all())->validate();
    event(new Registered(
        $user = $this->create($request->all())));
    //nonaktifkan kode di bawah ini
    //$this->guard()->Login($user);
}
```



View Register User

Untuk melihat tampilan login, pada web browser di url ketikkan
<http://localhost:8000/register>



The screenshot shows a web browser window with the address bar displaying `localhost:8000/register`. The page features a background illustration of a bookshelf filled with books. Overlaid on this is a white registration form titled "Register". The form contains four input fields: "Name", "E-Mail Address", "Password", and "Confirm Password". Below these fields is a blue button labeled "Register".



Membuat View Setelah Register User

Selanjutnya buatlah halaman afterRegister.blade.php dengan kode program sebagai berikut:

```
@section('content')
<div class="container">
<div class="jumbotron">
    <h2>Terima Kasih sudah Register</h2>
    <p>Mohon tunggu approval dari Administrator Kami</p>
</div>
</div>
@endsection
```



View Setelah Register User

Setelah Anda mengisi data-data user di form register, maka selanjutnya Anda di arahkan ke halaman afterRegister seperti tampak gambar di bawah ini:



Logging

- Logging merupakan sebuah proses yang mencatat informasi tentang segala peristiwa yang menyangkut kegiatan akses ke dalam sebuah server atau file.
- Proses logging akan merekam kegiatan user dalam sebuah sistem seperti usaha log in yang sah dan tidak sah, penggunaan sumber , aktivitas yang berkaitan dengan perubahan, penambahan, pengurangan dan penghapusan suatu file pada sistem yang tersimpan.
- Log aplikasi adalah segala catatan terkait dengan aplikasi kita. Catatan ini biasanya dituliskan oleh developer (atau otomatis ditulis oleh framework yang dipakai) untuk membantu proses **debugging** atau bug **fixing** di kemudian hari.
- Laravel secara default menyimpan log aplikasi di file **storage/logs/laravel.log**. Setiap error atau exception yang terjadi saat aplikasi berjalan akan disimpan di file tersebut.



Fungsi-Fungsi Logging di Laravel

```
Log::emergency($log);  
Log::alert($log);  
Log::critical($log);  
Log::error($log);  
Log::warning($log);  
Log::notice($log);  
Log::info($log);  
Log::debug($log);
```

Urutan fungsi-fungsi di atas menyatakan level seberapa penting log tersebut, dimulai dari **emergency** yang paling penting dan diakhiri dengan **debug** yang tidak terlalu penting. Fungsi apapun yang kita panggil, semua log akan ditulis ke dalam satu file yang sama, yaitu di file **laravel.log** yang berada di folder **storage/logs**.



Membuat Log::info

Untuk membuat **Log::info** informasi di Laravel, buka file **web.php** yang berada di folder **routes**, lalu tambahkan routing baru seperti kode di bawah ini:

```
use Illuminate\Support\Facades\Log;

Route::get('/log_view', function () {
    Log::info('Hello saya informasi dari log');
    return view('welcome');
});
```



Melihat Log::info

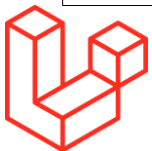
Setelah membuat routing baru untuk **Log::info**, silahkan jalankan server Laravel dengan perintah: **php artisan serve**. Lalu pada web browser di url ketikkan:
http://localhost:8000/log_view

Untuk melihat log informasi yang sudah dibuat, buka file **laravel.log** yang berada di folder **storage/logs**. Lalu lihat kode di akhir baris, maka kita bisa melihat Log:info yang sudah dibuat di dalam routing, seperti gambar di bawah ini:

```
storage > logs >  laravel.log
```

```
6232
```

```
6233 [2021-06-23 14:30:21] local.INFO: Hello saya informasi dari log
```



Membuat Log::debug

Untuk membuat **Log::debug** di Laravel, buka file **web.php** yang berada di folder **routes**, lalu tambahkan routing baru seperti kode di bawah ini:

```
use Illuminate\Support\Facades\Log;

Route::get('/log_debug', function()
{
    Log::debug('IP ' . Request::getClientIP() .
    'Hello saya Log::debug');
    return view('welcome');
});
```

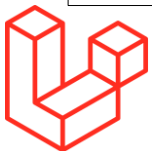


Melihat Log::debug

Setelah membuat routing baru untuk **Log::debug**, silahkan jalankan server Laravel dengan perintah: **php artisan serve**. Lalu pada web browser di url ketikkan: **http://localhost:8000/log_debug**

Untuk melihat Log::debug yang sudah dibuat, buka file **laravel.log** yang berada di folder **storage/logs**. Lalu lihat kode di akhir baris, maka kita bisa melihat Log:debug yang sudah dibuat di dalam routing, seperti gambar di bawah ini:

```
storage > logs >  laravel.log
6237    [2021-06-23 14:35:33] local.DEBUG: IP 127.0.0.1 Hello saya Log::debug
6238
```



**TERIMA KASIH
ATAS SEGALA PERHATIAN
SEMOGA BERMANFAAT...**

