MODUL 11
OTORISASI USER

CAPAIAN PEMBELAJARAN

1. Mampu mengatur Otorisasi User
2. Mengatur hak akases user

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. Sistem Operasi Linux/Window 10
2. PHP 5.x, diutamakan versi 7.x keatas
3. Composer
4. XAMPP (PHP, Apache, MySQL)
5. Text Editor Notepad++ atau Visual Studio Code

DASAR TEORI

1. Dasar Otorisasi

Selain menyediakan layanan autentikasi, Laravel juga menyediakan cara sederhana untuk mengotorisasi kewenangan *user* terhadap sumber daya yang diberikan. Seperti halnya autentikasi, Laravel untuk menggunakan pendekatan secara sederhana, dan ada dua cara untuk mengotori sasi tindakan: *Gate* dan *policies*.

Untuk pendekatan pintu masuk dan `route` dan `Controller`. `Gate` memberikan pendekatan yang sederhana berbasis *closure*, seperti controller. Sebagian besar aplikasi kemungkinan besar menggunakan otorisasi campuran `Gate` dan `Policies`. `Gate` bisa diterapkan untuk tindakan yang tidak terkait dengan model atau sumber daya apa pun, seperti melihat *dashboard administrator*. Otorisasi menggunakan `Gate` tempat yang sempurna untuk menerapkan logika. Sebaliknya,

policies harus digunakan ketika ingin mengotorisasi aksi/tindakan untuk model atau sumber daya tertentu.

2. Gates

Gates adalah kelas yang menentukan apakah user diberi kewenangan untuk melakukan tindakan tertentu atau tidak. Biasanya ditentukan dalam kelas `App\Providers\AuthServiceProvider` menggunakan fasad `Gate`. `Gate` selalu menerima *user* sebagai argumen pertama, dan dapat secara opsional menerima argumen tambahan seperti model `Eloquent`:

```
...
Gate::define('update-kota', function ($user, $kota) {
  return $user->id == $kota->user_id;
});
...
...
```

PRAKTIK

1. Menggunakan Gate

1) Cara menggunakan Gate, buka dan tambahkan dalam `app/Providers/AuthServiceProvider.php` seperti yang ditunjukkan pada kode program berikut:

```
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as
ServiceProvider;

class AuthServiceProvider extends ServiceProvider
{
    /**
     * The policy mappings for the application.
     *
     * @var array
     */
    protected $policies = [
```

```
        'App\Model' => 'App\Policies\ModelPolicy',
    ];

    /**
     * Register any authentication / authorization services.
     *
     * @return void
     */
    public function boot()
    {
        $this->registerPolicies();

        Gate::define('baca', function ($user, $kota) {
            return $user->id == $kota->user_id;
         });

    }
}
```

2) Kemudian buka dan tambahkan pada controller terkait, dalam hal ini file app/Http/Controllers/KotaController.php, seperti pada kode program berikut:

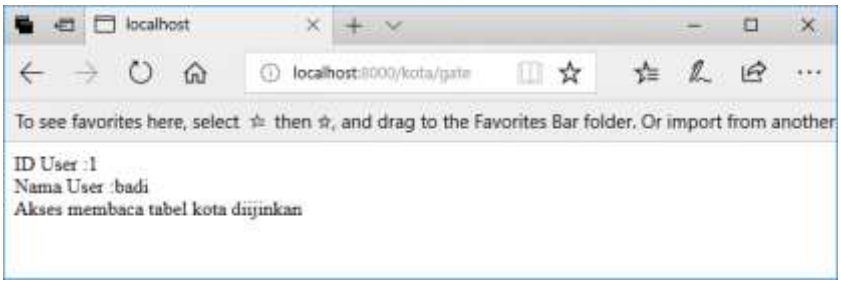
```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\models\Kota;
use Illuminate\Support\Facades\Gate;

class KotaController extends Controller
{

public function gate()
{
  $kota = Kota::find(1);

 if (Gate::allows('baca', $kota))
   {
    echo "ID User :".\Auth::user()->id."<br>";
    echo "Nama User :".\Auth::user()->name."<br>" ;
    echo "Akses membaca tabel kota diijinkan";
    }
 else
   {
     echo   "ID User :".\Auth::user()->id."<br>";
     echo   "Nama User :".\Auth::user()->name."<br>";
     echo "Akses membaca tabel kota tidak diijinkan";
   }
  exit;

 }
```

```
……..
……
}
```

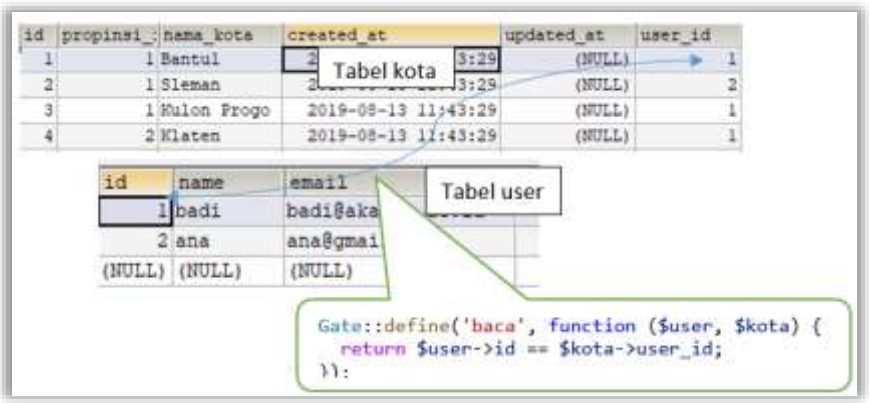
3) Berikut menambahkan `route` sehingga bisa memahami bagaimana *gate-based authorization* bekerja. Dalam `route` file `routes/web.php`, tambahkan `route` berikut:

```
Route::get('/kota/gate', 'KotaController@gate');
```

4) Cara menguji login dengan user yang sudah ada, lalu panggil url: http://localhost:8000/kota/gate



5) Setiap proses akan diperiksa ke kelas `Gate`, jika mengasilkan nilai benar maka akses diijinkan, seperti pada gambar berikut:



6) Coba uji login dengan *user* lain atau Anda ubah **`$kota = Kota::find(2);`** apa hasilnya?, Anda simpulkan

2. Menggunakan `Policies`

1) Untuk membuat `Policies` bisa menggunakan perintah *command line* artisan seperti pada perintah berikut:

```
php artisan make:policy KotaPolicy --model=Kota
```

Untuk argumen `--model=Kota` akan menambahkan beberapa method `view()`, `create()`, `update()`, `delete()`, di dalam `KotaPolicy.php`. Penggunaan Policies bisa membatasi akses perekaman, tergantung penerapannya.

2) Hasil dari pembuatan `KotaPolicy` terdapat di dalam subdirektori `app/Policies/PostPolicy.php`. Kemudian tambahkan seperti skrip program berikut:

```php
<?php

namespace App\Policies;

use App\User;
use App\models\Kota;
use Illuminate\Auth\Access\HandlesAuthorization;

class KotaPolicy
{
    use HandlesAuthorization;

    /**
     * Determine whether the user can view any kotas.
     *
     * @param  \App\User  $user
     * @return mixed
     */
    public function viewAny(User $user)
    {
        //
    }

    /**
     * Determine whether the user can view the kota.
     *
     * @param  \App\User  $user
     * @param  \App\models\Kota  $kota
     * @return mixed
     */
    public function view(User $user, Kota $kota)
    {
        return TRUE;
    }

    /**
     * Determine whether the user can create kotas.
     *
     * @param  \App\User  $user
     * @return mixed
     */
    public function create(User $user)
    {
```

```
        return $user->id > 0;
    }

    /**
     * Determine whether the user can update the kota.
     *
     * @param  \App\User  $user
     * @param  \App\models\Kota  $kota
     * @return mixed
     */
    public function update(User $user, Kota $kota)
    {
        return $user->id == $kota->user_id;
    }

    /**
     * Determine whether the user can delete the kota.
     *
     * @param  \App\User  $user
     * @param  \App\models\Kota  $kota
     * @return mixed
     */
    public function delete(User $user, Kota $kota)
    {
        return $user->id == $kota->user_id;
    }

    /**
     * Determine whether the user can restore the kota.
     *
     * @param  \App\User  $user
     * @param  \App\Kota  $kota
     * @return mixed
     */
    public function restore(User $user, Kota $kota)
    {
        return $user->id == $kota->user_id;
    }

    /**
     * Determine whether the user can permanently delete the kota.
     *
     * @param  \App\User  $user
     * @param  \App\Kota  $kota
     * @return mixed
     */
    public function forceDelete(User $user, Kota $kota)
    {
        return $user->id == $kota->user_id;
    }
}
```

Fungsi update:

```
public function update(User $user, Kota $kota)
    {
        return $user->id == $kota->user_id;
    }
```

Jika `$user->id` bernilai 1 dan `$kota->user_id` bernilai 1 (nilainya sama) akan menghasilkan nilai TRUE

3) Tambahkan pada App/Providers/AuthServiceProvider.php seperti kode program berikut:

```
?php

namespace App\Providers;

use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as
    ServiceProvider;
use App\models\Kota;
use App\Policies\KotaPolicy;

class AuthServiceProvider extends ServiceProvider
{
    /**
     * The policy mappings for the application.
     *
     * @var array
     */
    protected $policies = [
        'App\Model' => 'App\Policies\ModelPolicy',
        Kota::class => KotaPolicy::class
    ];
…..
}
```

4) Buatlah controller **KotaAutController.**php dan tambah seperti pada kode program berikut:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\models\Kota;
use Illuminate\Support\Facades\Auth;

class KotaAutController extends Controller
{
   public function view()
    {
        // membaca user login
        $user = Auth::user();

        // baca kota
```

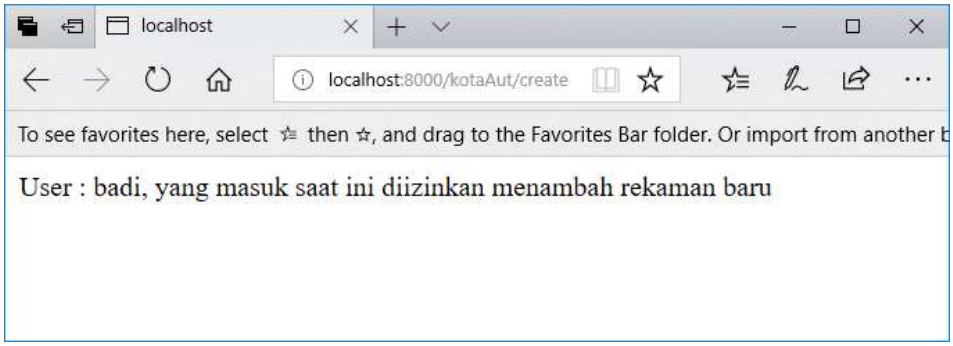
```
        $kota = Kota::find(1);

        if ($user->can('view', $kota)) {
          echo "Nama User ".$user->name.
          ", yang masuk saat ini diizinkan melihat rekaman Kota : ".
              $kota->nama_kota;
        } else {
           echo 'Tidak diijinkan  ->'.$user->name;
        }
    }

  public function create()
    {
        // membaca user login
        $user = Auth::user();

        if ($user->can('create', Kota::class)) {
          echo "User : ".$user->name.
            ", yang masuk saat ini diizinkan menambah rekaman baru";
        } else {
          echo 'Tidak diijinkan';
        }
      exit;
    }

    public function update()
    {
      // membaca user login
      $user = Auth::user();

      // baca kota
      $kota = Kota::find(1);

      if ($user->can('update', $kota)) {
        echo "Nama User ".$user->name.
        ", yang masuk saat ini diizinkan mengubah rekaman Kota : ".
         $kota->nama_kota;
      } else {
        echo "Tidak di ijinkan";
      }
    }

    public function delete()
    {
      // membaca user login
      $user = Auth::user();

      // baca kota
      $kota = Kota::find(1);

      if ($user->can('delete', $kota)) {
        echo "Nama User ".$user->name.
         ", yang masuk saat ini diizinkan meghapus rekaman Kota : ".
            $kota->nama_kota;
```

```
        } else {
          echo 'tidak dijinkan';
        }
    }
}
```

5) Tambahkan /routes/web.php, seperti pada kode program berikut:

```
…
Route::get('kotaAut/view','KotaAutController@view');
Route::get('kotaAut/create','KotaAutController@create');
Route::get('kotaAut/update','KotaAutController@update');
Route::get('kotaAut/delete','KotaAutController@delete');
…
```

6) Cara menguji, login dengan user tertentu lalu, jalankan url http://localhost:8000/kotaAut/create, hasilnya seperti pada gamabar berikut:



7) Lakukan pengujian lainnya:
   - http://localhost:8000/kotaAut/view
   - http://localhost:8000/kotaAut/update
   - http://localhost:8000/kotaAut/delete

LATIHAN

1. Diketahui tabel user dan tabel barang seperti pada gambar berikut:
   Tabel user

id	name	email	email_verifie	password
1	badi	badi@akakom.ac.id	(NULL)	$2y$10$qlzhxHZs06S73lurbNElZOnWcZ
2	ana	ana@gmail.com	(NULL)	$2y$10$7J5j2tsF216fQyjulu2/5ORO9E
3	Rangga	rangga@gmail.com	(NULL)	$2y$10$j//YhG5n9fy/Y3DsiQBOPuWKBP

Tabel barang

id	jenis_i	nama_barang	satuan	harga	stok	user_id
10001	1	Air Mineral	Botol	5000	7	1
10002	2	Minyak Goreng	Pcs	20000	16	1
10003	1	Penggaris	Batang	5000	100	1
10004	1	Penghapus	Pcs	4000	29	1
10005	1	Buku Bergaris	Lb	5000	8	1
10007	1	Spidol	Batang	5000	200	1
10008	2	Gula Putih	Kg	10000	10	1
10009	1	Buku Gambar	Pcs	10000	10	1
10010	1	Buku Bacaan	Pcs	10000	10	1
10011	1	Kartas A 4	Rem	30000	2	1
10012	1	Buku Gambar Mewarn	Pcs	10000	10	1
10013	2	Kopi Mix	Saset	2500	100	1
10020	1	Buku Tulis	Pcs	10000	10	1

2. Buatlah otorisasi mengunakan `Gate` yang boleh mengubah barang user dengan user_id=1 saja

3. Buatlah otorisasi mengunakan `Policies` yang boleh mengubah barang user dengan user_id=1 saja

TUGAS

1. Jelaskan skrip di bawah ini

```
[1]   <?php
[2]   namespace App\Http\Controllers;

[3]   use App\Http\Controllers\Controller;
[4]   use App\Post;
[5]   use Illuminate\Support\Facades\Gate;

[6]   class PostController extends Controller
[7]   {

[8]   public function gate()
[9]   {
[10]  $post = Post::find(1);

[11]  if (Gate::allows('update-post', $post)) {
[12]  echo 'Allowed';
[13]  } else {
```

```
[14]   echo 'Not Allowed';
[15]   }

[16]   exit;
[17]   }
[18]   }
```

2. Buktikan User dan dengan model Post

REFERENSI

Sajal Soni, 2017, Gates and Policies in Laravel, https://code.tutsplus.com/tutorials/gates-and-policies-in-laravel--cms-29780