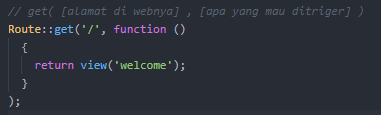
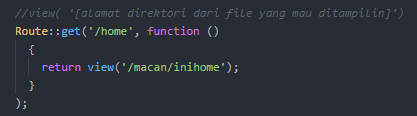
# Laravel Dasar

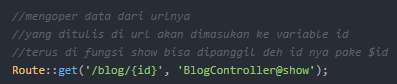
## Routes and View

Routes untuk mengatur alur suatu alamat url ke file tampilan (view) atau ke controller









### View dan Redirect

Fucntion View menuju ke folder views. Parameter 1 : alamat , Parameter 2 : melempar nilai atau variable



Function redirect menuju ke url yg tercantum di routes.



## Controller and Blade

Kenapa alurnya harus lewat controller dulu?  
Agar diolah dulu datanya di controller, pengolahan yang udah dipelajari yaitu

### Mengoper data

File route :

Route::get('/blog/{id}', 'BlogController@show');

File controller :

public function show ($id)

{

$hasil = $id;

Trivia :

$users = ['Satria Laksana','Vitrah Raisa'];

$unescaped = '<script>alert("isi dari varible $unescaped di eksekusi")</script>';

// oper lebih dari 1 variable ke halaman tujuan

return view('bloggy/iniblog' , ['nilai' => $hasil,'users' => $users,'unescaped' => $unescaped]);

}

File View :

{{$nilai}}

Syntax ini sama dengan echo $nilai;

{!! $unescaped !!}  
// unescaped berfungsi untuk memperlakukan isi dari variable $unescaped sebagai script yang akan di eksekusi.

### contoh code penggunaan blade template.

@if (count($users) > 5)

<p>User lebih dari lima</p>

@else

<p>User tidak lebih dari lima</p>

@endif

Note :   
count() digunakan untuk menghitung jumlah kolom array didalam variable $users.

@foreach ($users as $user)

<li>{{$user->username}}</li>

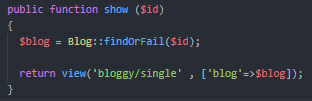
@endforeach

### Membedakan url /blog dan /blog/{id}

/blog diarahkan ke BlogController@index sedangkan /blog/{id} diarahkan ke BlogController@show  
note : index dan show adalah nama function didalam class yang biasanya digunakan untuk membuat pengolahan halaman umum dan halaman khusus.

#### Error Message yang Ramah

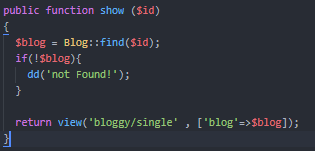
##### Find or Fail



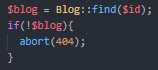
Tampilan ini muncul karena tidak ada file 404.blade.php nya

Jika fail, ia akan membuka file views/errors/404.blade.php

##### Die dump



##### abort



Abort akan menuju ke file angka parameter di dalam folder views/errors.

### RESTful Resource Controller

Resource controllers make it painless to build RESTful controllers around resources. For example, you may wish to create a controller that handles HTTP requests regarding "photos" stored by your application. Using the make:controller Artisan command, we can quickly create such a controller:

php artisan make:controller PhotoController

Next, we register a resourceful route to the controller:

Route::resource('photo', 'PhotoController');

This single route declaration creates multiple routes to handle a variety of RESTful actions on the photo resource. Likewise, the generated controller will already have methods stubbed for each of these actions, including notes informing you which URIs and verbs they handle.

#### Actions Handled By Resource Controller

| **Verb** | **Path** | **Action** | **Route Name** |
| --- | --- | --- | --- |
| GET | /photo | index | photo.index |
| GET | /photo/create | create | photo.create |
| POST | /photo | store | photo.store |
| GET | /photo/{photo} | show | photo.show |
| GET | /photo/{photo}/edit | edit | photo.edit |
| PUT/PATCH | /photo/{photo} | update | photo.update |
| DELETE | /photo/{photo} | destroy | photo.destroy |

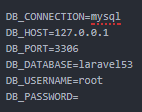
## Sistem Templating Blade

Dengan sistem ini, kita ga harus nulis kodingan yang sama berulang kali, misalnya bagian header sama footer kan selalu sama disetiap halaman, maka tinggal buat satu file dan terus diextend ke halaman yang mmau kita buat.

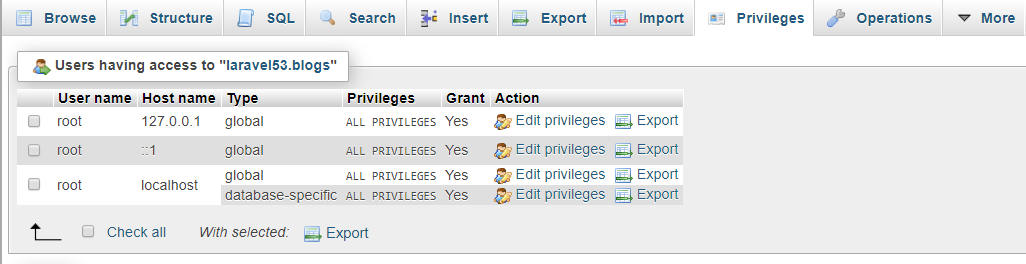
## Database dengan Query Builder

### Setting awal

Harus disetting terlebih dahulu file .env di root folder nama databasenya, username dan passwordnya



Username dan password taunya dari mana?

Kalo masih bersifat lokal, username dan passwordnya adalah root, dan di phpMyAdmin juga ada infonya di tab privileges, tapi entah dikasus aku passwornya harus dikosongin baru bisa.

Kalo udah di hosting, itu gampang liat di Cpanelnya.

### Pengaplikasian di kode

Di file controllernya dikasih :  
use Illuminate\Support\Facades\DB;

Terus untuk ngambil datanya tinggal pake syntax :   
$users = DB::table('users')->get();

Note : harus ditentuin table mana yang mau diambil. Nanti variable $users akan bertype objek (maksudnya nanti harus dideklarasiin lagi kolom mana yang mau diambil)

Untuk di file viewnya, tinggal output pake perulangan foreach :

@foreach ($users as $user)

<li>{{$user->username}}</li>

@endforeach

### Troubleshooting

#### Masalah ga bisa konek ke database

Error Message :  
"SQLSTATE[HY000] [1045] Access denied for user 'root'@'localhost' (using password: YES) (SQL: select \* from `nama`)"

Hipotesa :  
Salah username atau password di file .env

my case :  
salah password, di isi "root" ga bisa-bisa ketika password di kosongkan

solution :  
kosongin password di file .env  
matiin nyalain lagi "php artisan serve"  
done

## Eloquent

### Model

Model ini dia bertanggung jawab buat suatu 1 table khusus didalam database, jadi ga harus define table lagi kalo pake model.

##### Cara Buat

php artisan make:model [nama modelnya]

##### Pengaturan

secara default class blog akan mengurus table yang bernama 'Blogs' (bentuk plural dari nama classnya)   
tapi kalo nama table kita bukan 'Blogs', maka kita bisa ngasih tahu program dengan syntax :

protected $table = 'blog123';

Model punya fitur mencatat tgl pembuatan dan update dari tsb, tapi kalo kita ga punya kolom create\_at dan update\_at, maka fitur ini bisa dimattin, cara :

public $timestamps = false;

##### Cara pakai

Di file controllernya kasih :  
use App\Blog;

Terus didalem classnya :

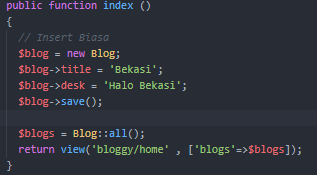
$blogs = Blog::all();  
dd($blogs);

Note :   
all() adalah metode untuk mengambil semua row di dalam table.  
dd([variable]) adalah metode die dump, untuk menampilkan isi dari varible dan mengabaikan proses setelahnya.

## CRUD Eloquent

### Insert

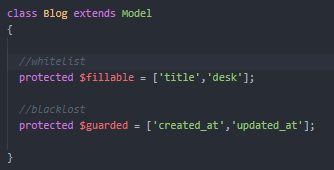
#### Cara Biasa



#### Mass Assigment



Pada mass asigment, di file model harus di set whitelist atau blacklistnya

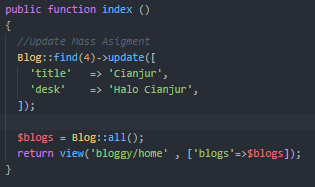


### Update

#### Biasa

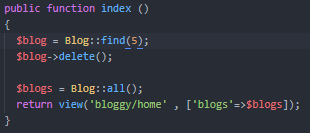


#### Mass Asigment

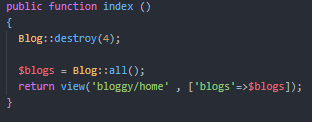


### Delete

#### Biasa



#### destroy()



destroy bisa menghapus beberapa data sekaligus

#### Soft Deletes

##### Preparation

phpMyAdmin :

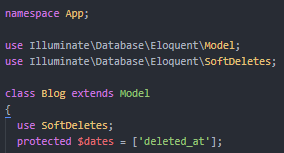
Buat coloumn deleted\_at



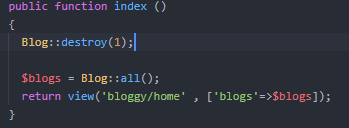


Model :

Aktifkan use softdeletes pada namespace dan juga didalam classnya.



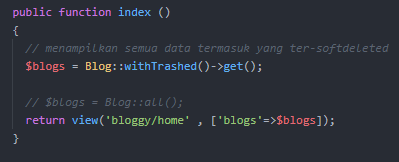
##### Menghapus



Sama seperti menghapus biasa, namun row di database tidak terhapus, hanya mengupdate coloumn deleted\_at.

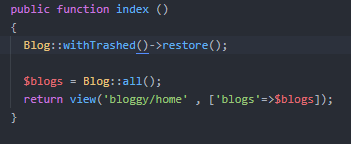
##### Trash

Row yang ter-SoftDeleted disebut *trash* , cara menampilkan trash :



##### Restore

Restore mengembalikan nilai coloumn deleted\_at kembali ke NULL.



##### Trivia

Untuk reset auto\_increment :

ALTER TABLE table\_name AUTO\_INCREMENT = 1;

Note : ketik di console phpMyAdmin, Ctrl + Enter

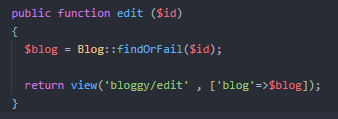
## Update by User

File route :

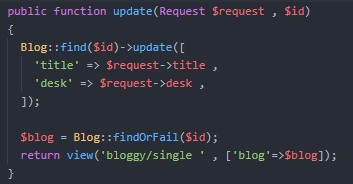


File Controller :

Buat fungsi edit.

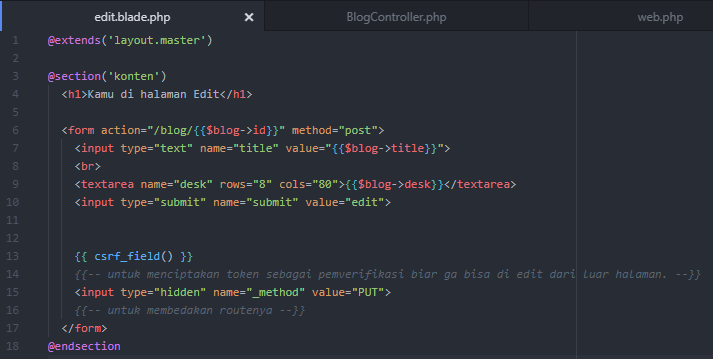


Buat juga fungsi update.



File view :

Buat file edit.blade.php



Alur :

Route::get('/blog/{id}/edit', 'BlogController@edit');

function edit

edit.blade.php

Route::put('/blog/{id}', 'BlogController@update');

Function update

## Delete by User

1. Buat form delete

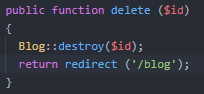




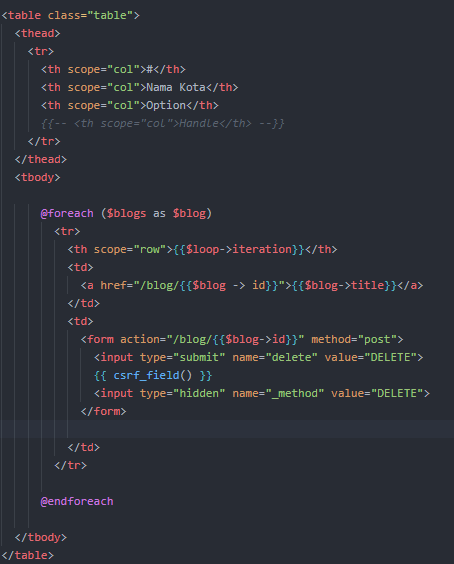
1. Buat route ke controller

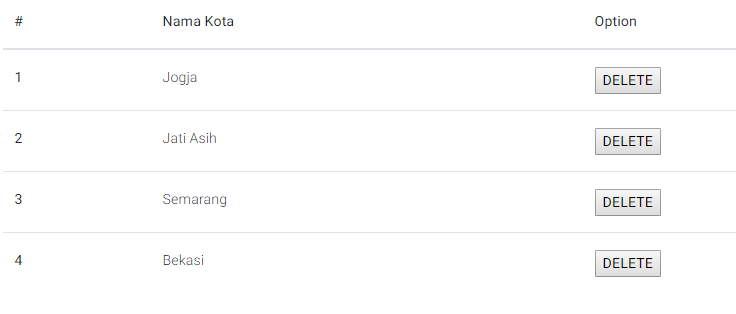


1. Buat fungsi di controllernya



## Display database into table





### The $loop variable

The $loop variable also contains a variety of other useful properties:

| **Property** | **Description** |
| --- | --- |
| $loop->index | The index of the current loop iteration (starts at 0). |
| $loop->iteration | The current loop iteration (starts at 1). |
| $loop->remaining | The iterations remaining in the loop. |
| $loop->count | The total number of items in the array being iterated. |
| $loop->first | Whether this is the first iteration through the loop. |
| $loop->last | Whether this is the last iteration through the loop. |
| $loop->even | Whether this is an even iteration through the loop. |
| $loop->odd | Whether this is an odd iteration through the loop. |
| $loop->depth | The nesting level of the current loop. |
| $loop->parent | When in a nested loop, the parent's loop variable. |

## Validasi Laravel

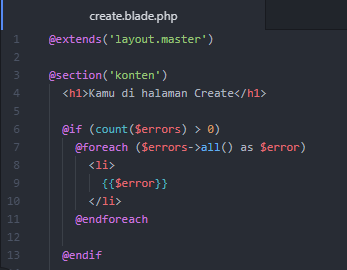
Validasi mengatur biar user inputnya ga asal-asalan.



### Menampilkan errornya di page :

Kalo ada syarat yg ga terpenuhi, laravel udh otomatis buat variable $errors yang tinggal dimunculin pake if.

#### Dengan foreach



#### Pengaturan posisi



### Menyimpan nilai sebelumnya

Pada title :



Pada text area :



## Migration

### Fungsi :

1. Buat otomatis bikin table yang dibutuhin project kita, in case kita mau sharing ke temen.
2. Bisa undo kalo bikin kesalahan.

### Generating Migrations

php artisan make:migration [nama file migrationnya] --create=[nama table yang akan dibuat]

Contoh :

php artisan make:migration create\_users\_table --create=users

### Run

php artisan migration

#### Drop All Tables & Migrate

The migrate:fresh command will drop all tables from the database and then execute the migrate command:

php artisan migrate:fresh  
php artisan migrate:fresh –seed

### Adding Columns

To update an existing table, we will use the Schema::table method:

Schema::table('users', function($table)

{

$table->string('email');

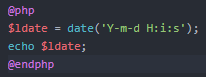
});

The table builder contains a variety of column types that you may use when building your tables:

| **Command** | **Description** |
| --- | --- |
| $table->bigIncrements('id'); | Incrementing ID using a "big integer" equivalent |
| $table->bigInteger('votes'); | BIGINT equivalent to the table |
| $table->binary('data'); | BLOB equivalent to the table |
| $table->boolean('confirmed'); | BOOLEAN equivalent to the table |
| $table->char('name', 4); | CHAR equivalent with a length |
| $table->date('created\_at'); | DATE equivalent to the table |
| $table->dateTime('created\_at'); | DATETIME equivalent to the table |
| $table->decimal('amount', 5, 2); | DECIMAL equivalent with a precision and scale |
| $table->double('column', 15, 8); | DOUBLE equivalent with precision, 15 digits in total and 8 after the decimal point |
| $table->enum('choices', ['foo', 'bar']); | ENUM equivalent to the table |
| $table->float('amount'); | FLOAT equivalent to the table |
| $table->increments('id'); | Incrementing ID to the table (primary key) |
| $table->integer('votes'); | INTEGER equivalent to the table |
| $table->json('options'); | JSON equivalent to the table |
| $table->jsonb('options'); | JSONB equivalent to the table |
| $table->longText('description'); | LONGTEXT equivalent to the table |
| $table->mediumInteger('numbers'); | MEDIUMINT equivalent to the table |
| $table->mediumText('description'); | MEDIUMTEXT equivalent to the table |
| $table->morphs('taggable'); | Adds INTEGER taggable\_id and STRING taggable\_type |
| $table->nullableTimestamps(); | Same as timestamps(), except allows NULLs |
| $table->smallInteger('votes'); | SMALLINT equivalent to the table |
| $table->tinyInteger('numbers'); | TINYINT equivalent to the table |
| $table->softDeletes(); | Adds **deleted\_at** column for soft deletes |
| $table->string('email'); | VARCHAR equivalent column |
| $table->string('name', 100); | VARCHAR equivalent with a length |
| $table->text('description'); | TEXT equivalent to the table |
| $table->time('sunrise'); | TIME equivalent to the table |
| $table->timestamp('added\_on'); | TIMESTAMP equivalent to the table |
| $table->timestamps(); | Adds **created\_at** and **updated\_at** columns |
| $table->rememberToken(); | Adds remember\_token as VARCHAR(100) NULL |
| ->nullable() | Designate that the column allows NULL values |
| ->default($value) | Declare a default value for a column |
| ->unsigned() | Set INTEGER to UNSIGNED |

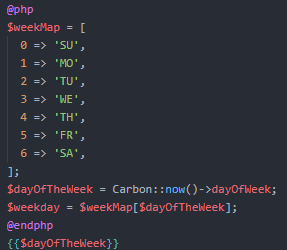
## Date

### tanggal



Output : 2019-04-14 15:33:47

### Hari



Output : SU