Page

Overview Laravel

Web Prog

> **di baca aja, nanti ada pdf 1 lagi bahas crud 1 project**

Rangkuman

- Create Project , Setup project? ENV
- Views → Laravel templating engine
- Model → Migration→ Seeder and Relation model
- Routing
- Controller

Create Project

Bash

```
composer create-project laravel/laravel namaAppKalian
```

Jalanin Project

Bash

```
cd namaAppKalian

php artisan serve
```

How to setup your project?

dalam .env

yang harus di perhatikan yaitu

Text

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=namaDatabasekelean
DB_USERNAME=root
DB_PASSWORD=
```

ada case khusus jika kalian pull → ini hanya berlaku jika kalian install project langsung yaa

oiya selalu pastiin server kalian jalan yaa xampp (apache mysql) / penyedia services lainnya dan artisan servenya

---

Views

cara membuat views? **namafile.blade.php** contoh main.blade.php , kenapa harus pake blade.php? karena kita mau menggunakan kemampuan dari templating enginenya laravel

ada beberapa hal yang harus kalian ketahui yaitu @include, @extends, @section & @endsection, @yields

@include('namafile')

- itu buat include sebuah views ke dalam sebuah views di blade

@extends('namafile parent')

- nandain bahwa sebuah file itu sebagai turunan/extend dari file apa

@section('namasection') & @endssection

- digunakan sebagai lingkup section yang mau kita extends/ masukkan

@yield('namasection)

- digunakan untuk memanggil section yang kita gunakan

Contoh Penggunaanya

main.blade.php

```php
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>@yield('title')</title>
</head>

<body>

    @include('partials.navbar')

    <div class="container mt-4">
        @yield('container')
    </div>

</body>

</html>
```

di dalam folder partials yang sejajar dengan main.blade.php ada file navbar.blade.php

navbar.blade.php

```php
<nav class="navbar navbar-expand-lg navbar-dark">
  ========== code navbar kalian ==========
</nav>
```

home.blade.pphp

```php
@extends('main')

@section('container')
    <h1> Halaman Home</h1>
@endsection
```

FAQ

- include bisa dipakai banyak ga? bisa
- yields dan section bisa di pakai banyak dengan nama yang berbeda? bisa
  - contohnya gimana ka?

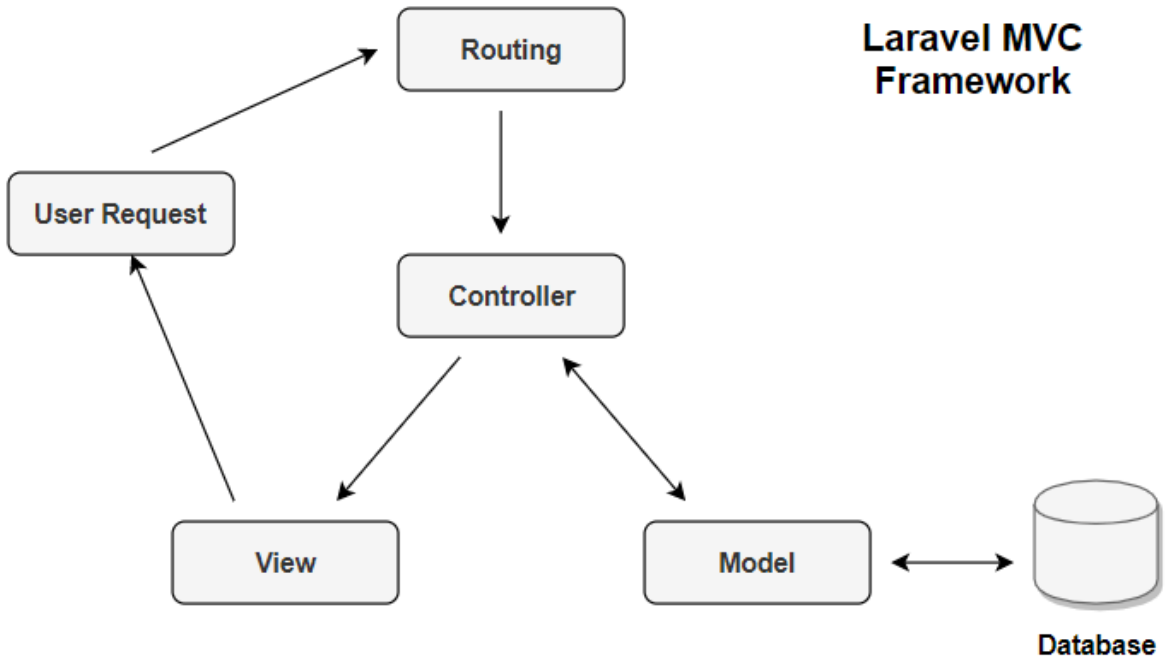
    ```php
    <title>{{ $title }}</title>
    ```

    ```php
    @extends('main')

    @section('title', 'Page Title')

    @section('container')
        <h1> Halaman Home</h1>
    @endsection
    ```

- kalau nda ada extends gimana? error harusnya coba aja hihih

---

Model



Dalam pembuatan sebuah website biasanya kita membutuhkan sebuah database. Model dalam laravel punya architecture yang bertugas untuk menghubungkan project laravel kita dengan database.

sebelum masuk kedalam model lebih baiknya kita tahu beberapa hal terlebih dahulu

- setup env → database✅
- migration?
- model?
- seeder?
- factory?

---

apa itu migration?

sebuah fungsi dari laravel yang memungkinkan kita membuat struktur sebuah table di database. fungsinya apa? supaya dalam development nanti kita bisa memiliki schema database yang sama dengan tim tanpa membuat satu tablenya

gimana cara membuatnya?

Bash

```
php artisan make:migration create_books_table
```

PHP

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('book', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->string('slug')->unique();
            $table->string('image')->nullable();
            $table->text('description');
            $table->timestamp('published_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('book');
    }
};
```

cara menjalankan migrasi?

Bash

```bash
php artisan migrate
```

atau

Bash

```bash
php artisan migrate:fresh
```

---

apa itu model?

model merupakan sebuah file/architecture laravel yang memungkinkan kita untuk memperkenalkan laravel dengan table di database kita

```bash
php artisan make:model Book
```

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Book extends Model
{
    use HasFactory;
}
```

basic apa yang harus kalian tahu? fillable, guarded, dan cara membuat relationship

```php
    protected $fillable = ['title','body','description'];

    // protected $guarded = ['id'];
```

fillable → seperti namanya yang berkaitan dengan fill. intinya fillable ini digunakan untuk menginformasikan laravel kalau fild atau column itu boleh di

guarded? sama tapi dia menjaga, idnya supaya tidak bisa di isi

pakai yang mana? salah satu aja...

```php
protected $table = 'my_flights'; // nama table kalian
```

define primary key kalian → berguna pas primary key kalian bukan id

```php
protected $primaryKey = 'flight_id';
```

gimana cara membuat relationshipnya?

step pertama

1. pastikan kalian sudah membuat struktur table yang relation accepted contoh

```php
Schema::table('blogs', function (Blueprint $table) {
    $table->foreign('kategoriId')->references('id')->on('categories');
    $table->foreign('writerId')->references('id')->on('users');
});
```

tips → kita bisa loh menambahkan relasi dengan membuat migration baru hihih. sebagai contoh sudah terdapat migration untuk table blogs, tapi lupa membuat referencesnya? yowes buat migration lagi saja

```php
$table->foreign('kategoriId')->references('id')->on('categories');
```

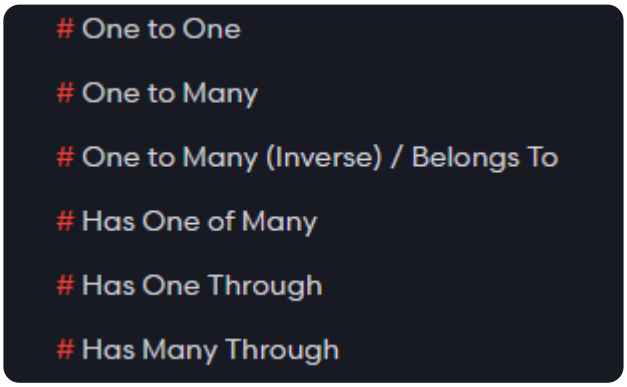
memiliki arti , laravel tolong buatin relasi untuk kategorild yang reference id di table categories

tips → pastikan si table referencenya tuh udah dibuat ya dan memiliki id yang tepat

Laravel - The PHP Framework For Web Artisans
https://laravel.com/docs/11.x/eloquent-relationships#main-content

setelah itu tandain di model kalian dengan

ada macem" ada one to one, one to many, many to one, dkk



one to one

user model

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasOne;

class User extends Model
{
    /**
     * Get the phone associated with the user.
     */
    public function phone(): HasOne
    {
        return $this->hasOne(Phone::class);
    }
}
```

phone model

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Phone extends Model
{
    /**
     * Get the user that owns the phone.
     */
    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }
}
```

one to many / many to one?

post model

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Post extends Model
{
    /**
     * Get the comments for the blog post.
     */
    public function comments(): HasMany
    {
        return $this->hasMany(Comment::class);
    }
}
```

comments

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Comment extends Model
{
    /**
     * Get the post that owns the comment.
     */
    public function post(): BelongsTo
    {
        return $this->belongsTo(Post::class);
    }
}
```

sebagai contoh

Model Post

```php
    public function category(){
        return $this->belongsTo(Category::class);
    }
```

since dia ada di model post maka 1 post, 1 category

Model Category

```php
public function posts(){
    return $this->hasMany(Post::class);
}
```

1 category, bisa banyak post

sometimes dalam development ada error/ tidak sesuai proses bisnis kita

sebagai guide selalu tandain id apa yang jadi referencenya

dalam model post

```php
  public function author(){
      return $this->belongsTo(User::class, 'user_id');
  }
```

artinya apa? dalam 1 post ada 1 user yang mana di tandai dengan user_id

---

apa itu seeder?

fungsi di laravel yang memudahkan kita untuk mengisi/mengseeding database kita (input otomatis)

Bash

```
php artisan make:seed userSeeder
```

PHP

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;

class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeders.
     */
    public function run(): void
    {
        DB::table('users')->insert([
            'name' => Str::random(10),
            'email' => Str::random(10).'@example.com',
            'password' => Hash::make('password'),
        ]);
    }
}
```

disclaimber dengan catatan databaseSeeder.php sudah di panggil ya class seedernya

seperti dibawah ini

PHP

```php
<?php

namespace Database\Seeders;

use App\Models\User;
use App\Models\Blog;
use Database\Factories\BlogFactory;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Database\Seeders\CategorySeeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        User::factory(2)->create();

        $this->call([ // <- seperti ini
            CategorySeeder::class // <- seperti ini
        ]); // <- seperti ini
        Blog::factory(10)->create();
        // User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);
    }
}
```

bagaimana menjalankan nya? wait ya

PHP

```php
php artisan db:seed
```

specific seeder

PHP

```php
php artisan db:seed --class=namafile
```

tips → kalian bisa loh langsung membuat model, migration dan seeder

Text

```
php artisan make:model NamaModel -m -s
```

mengclear schema database + seeding

Text

```
php artisan migrate:fresh --seed
```

loh kok itu ada Blog::factory(10)->create(); ? apa itu?

factory simplenya kita sudah membuat sebuah template buat ngisi juga sama kayak seeder dengan fitur yang lebih advance dan cepet

cara buatnya?

PHP
```
php artisan make:factory namaFactory
```

PHP
```
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Blog>
 */
class BlogFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            'title' => $this->faker->sentence(mt_rand(2,8)),
            'slug' => $this->faker->slug(),
            'image' => 'default.png',
            'body' => collect($this->faker->paragraphs(mt_rand(5,10)))
                      ->map(fn($p) => "<p>$p</p>")
                      ->implode(''),
            'writerId' => mt_rand(1,1),
            'kategoriId' => mt_rand(1,1),
         ];
    }
}
```

kalian bisa belajar pake faker juga (explore sendiri ya)

FakerPHP / Faker
https://fakerphp.org/

Apa itu Routing?

Routing bisa di bilang adalah sebuah fitur yang memungkinkan kita untuk membuat arah jalan. Analoginya kayak gerbang tol yang memiliki cabang ke arah yang berbeda masing" pintu gerbang tolnya

[tol pamulang]

Routing biasanya bisa digunakan dengan berbagai macam seperti, menampilkan views, melakukan operasi create, update, delete dan macam"

basic routing itu seperti ini

```php
use Illuminate\Support\Facades\Route;

Route::get('/greeting', function () {
    return 'Hello World';
});
```

meaning dari code diatas adalah ketika kita mengakses link/url '/greeting' laravel bakal ngembalikan views 'Hellow World'

penggunaan selanjutnya view

```php
Route::view('/welcome', 'welcome');

Route::view('/welcome', 'welcome', ['name' => 'bambang']);
```

disini kita menugaskan route **view** untuk menampilkan sebuah halaman welcome.blade.php

untuk yang line ke 3, artinya kita membawa sebuah array sosiatif 'name' dengan value bambang

kalau kalian notice ada route::get, ada route::view ada yang lain nda? ada

secara garis besar ada macam" route method yaitu

```php
Route::get($uri, $callback); -> untuk mempasing data, atau menampilkan views
Route::post($uri, $callback); -> untuk mengcreate data
Route::put($uri, $callback); -> untuk mengedit data
Route::patch($uri, $callback); -> untuk mengpatch -> tidak terlalu dipakai
Route::delete($uri, $callback); -> untuk mendelete data
```

$callback → biasanya diisi dengan sebuah fungsi, biasanya fungsi biasa atau function

fungsi biasa bisa mengembalikan views

```php
Route::get('/about', function () {
    return view('About', [
        "title" => "About"
    ]);
}
```

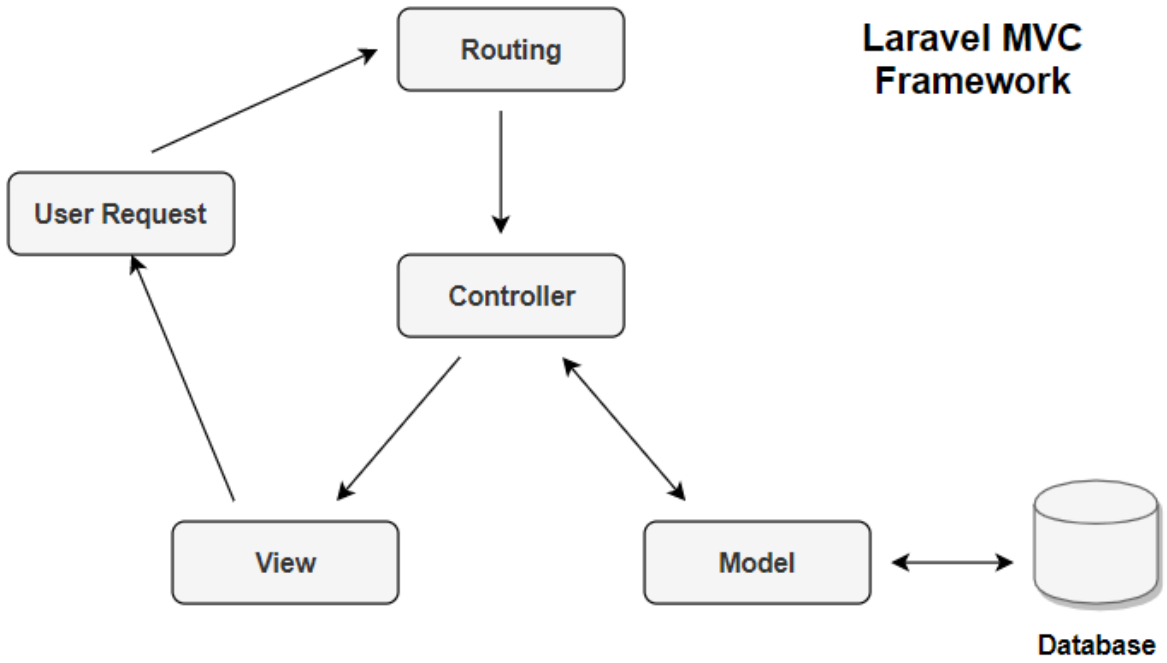
atau sebuah controller

```php
use App\Http\Controllers\UserController;

Route::get('/user', [UserController::class, 'index']);
```

untuk melihat daftar route yang kita punya

```bash
php arttisan route:list
```

Apa itu controller?



controller merupakan sebuah perantara antara views dan model

controller biasanya isinya fungsi" yang bakal di akses di routes kita. Jadi ada korelasi seperti gambar diatas.

untuk membuat controller

```bash
php artisan make:controller BookController
```

Create

kalau kita mau create suatu data yang harus kita siapin yaitu

- method create buat nampilin views create
- pastikan setiap inputan memiliki name yang kalian tau
- menambahkan method="post", action="" dan @csrf didalam form serta enctype="multipart/form-data" jika terdapat inputan file/image didalam form kita (button dengan type submit jg harus yaa)
- method store buat menyimpan datanya

step 1 : gimana cara buat nampilin views create melalui controller?

```php
    public function create()
    {
        return view('books.create');
    }
```

apa arti code diatas? artinya "laravel tolong kalau ada orang akses method create di controllerr [nama controller] balikin views di folder books yang namanya create"

jika sudah set route untuk masuk ke controller dan method create tersebut

PHP
```
Route::get('/create',[BooksController::class,'create']);
```

step 2 : memastikan setiap inputan memiliki name

PHP
```
<div class="mb-3">
    <label for="title" class="form-label">Title</label>
    <input type="text" class="form-control" id="title" name="title" required
autofocus>
</div>
```

seperti contoh diatas

step 3: selanjutnya adalah menambahkan method="post", action="" dan @csrf didalam form serta enctype="multipart/form-data"

pastikan actionnya ke method path yang sudah dibuat di route yaa

PHP
```
Route::get('store',[BooksController::class,'store']);
```

```
<form method="post" action="/store" class="mb-5" enctype="multipart/form-data">
    @csrf
    <div class="mb-3">
        <label for="title" class="form-label">Title</label>
```

step 4: menyiapkan method store

PHP
```
    public function store(Request $request)
    {

      $data = [
          'title' => $request->input('title'),
          'author' => $request->input('author'),
          'description' => $request->input('description', 'Default description'), //
Provide a default value if description is not provided
      ];

      Book::create($data);

    return redirect()->route('books.index');
    }
```

PHP
```
$data = [
          'title' => $request->input('title'),
          'author' => $request->input('author'),
          'description' => $request->input('description'),
      ];
```

artinya kita menyimpan data dari request user dan memindahkannya ke array

yang harus di perhatikan input('title') itu referencenya value dari inputan yang punya name title

```php
Book::create($data);
```

dengan menggunakan bantuan model dan query builder kita memasukkan data ke database dengan cara

namamodel::create($data)

```php
    return redirect()->route('create');
```

artinya kita redirect ke /create

disclaimer pastikan $data harus sama ya kayak fillable di model sebagai contoh

kalau kalian di model adanya seperti ini

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Book extends Model
{
    use HasFactory;
    protected $table = 'books'; // nama table kalian
    protected $primaryKey = 'id';
    protected $fillable = ['title','description'];
}
```

maka si controllernya harus bergini

```php
    public function store(Request $request)
    {

      $data = [
          'title' => $request->input('title'),
          'description' => $request->input('description', 'Default description'), //
Provide a default value if description is not provided
      ];

      Book::create($data);

    return redirect()->route('books.index');
    }
```

```php
public function store(Request $request)
```

yang perlu di perhatikan Request $request di dapatkan ketika ada request lewat form ketika submit

Read

biasanya kalau sederhana menampilkan data, ada 2 tipe

- menampilkan banyak data sekaligus
- menampilkan data sekaligus

kalau menampilkan banyak data sekaligus yang harus kita siapin yaitu

- controller buat menampilkan viewsnya

```php
public function index()
{
    return view('index');
}
```

- menyiapkan routenya

```php
Route::get('/books', [BooksController::class,'index']);
```

- menyiapkan data yang dipasing di controller

```php
public function index()
{
    $books = Book::all(); -> banyak opsi lain
    return view('index');
}
```

dalam case ini banyak opsi untuk menampilkan data

contoh

```php
public function index()
{
    $books = Book::select('title', 'author')->get(); cuman column title author
    return view('index');
}
```

```php
public function index()
{
     $keyword = 'Bad Habits'
     $books = Book::where('author', 'John Doe')
                    ->where('title', 'like', '%' . $keyword . '%')
                    ->orderBy('reviews_count', 'desc');
    return view('index');
}
```

Laravel - The PHP Framework For Web Artisans
https://laravel.com/docs/11.x/queries#introduction

- mempasing datanya ke views

```php
public function index()
{
    $books = Book::all();
    return view('index', 'books' => $books);
}
```

- menampilkannya data dalam views dengan foreach (biasanya)

```php
<!DOCTYPE html>
<html>
<head>
    <title>Books</title>
</head>
<body>
    <h1>Books</h1>
    <table>
        <tr>
            <th>Title</th>
            <th>Author</th>
            <th>Publication Date</th>
            <th>Reviews Count</th>
        </tr>
        @foreach ($books as $book)
        <tr>
            <td>{{ $book->title }}</td>
            <td>{{ $book->author }}</td>
            <td>{{ $book->publication_date }}</td>
            <td>{{ $book->reviews_count }}</td>
        </tr>
        @endforeach
    </table>
</body>
</html>
```

menampilkan 1 data spesifik

- siapkan controllernya

```php
 public function show(Book $book)
{

}
```

- ambil data sesuai request

```php
 public function show(Book $book)
{
     return view('show',[
        'book' => $book
    ]);
}
```

- menampilkan views

```php
{{$book->title}}
```

sesuaikan dengan atribute field yang ada di migration

- route untuk show

```php
Route::get('/book/{id}',[BlockController::class,'show']);=
```

penggunaanya ketika

```php
<!DOCTYPE html>
<html>
<head>
    <title>Books</title>
</head>
<body>
    <h1>Books</h1>
    <table>
        <tr>
            <th>Title</th>
            <th>Author</th>
            <th>Publication Date</th>
            <th>Reviews Count</th>
            <th>Actions</th>
        </tr>
        @foreach ($books as $book)
        <tr>
            <td>{{ $book->title }}</td>
            <td>{{ $book->author }}</td>
            <td>{{ $book->publication_date }}</td>
            <td>{{ $book->reviews_count }}</td>
            <td><a href="/book/{{$book->id}}">Show Details</a></td>
        </tr>
        @endforeach
    </table>
    {{ $books->links() }} <!-- Pagination links -->
</body>
</html>
```

Update

analoginya sama kayak show dan create yaitu dapetin data sebelumnya lalu baru ada operasi buat update

- siapkan method edit untuk menampilkan data serta viewsnya

```php
public function edit(Post $post)
{
    return view('edit',[
        'post' => $post
    ]);
}
```

- siapkan viewsnya yang menampilkan data sebelumnya → (replace aja dari create) tambahkan value

```php
<input type="text" id="title" name="title" value="{{ $post->title) }}">
```

- method post , action dan ectype serta @method('put') @csrf

```
<form method="post" action="/book/{{ $book->id }}" class="mb-5" enctype="multipart/form-data">
    @method('put')
    @csrf
    <div class="mb-3">
```

- siapin method updatenya

```php
public function update(Request $request, Book $book)
    {
      $data = [
        'title' => $request->input('title'),
        'author' => $request->input('author'),
        'description' => $request->input('description')
      ];

       Book::where('id',$post->id) ->update($data);
       return redirect('/index');
    }
```

- siapin routesnya

```php
Route::get('/book/{id}',[BookController::class,'edit']);
Route::put('/book/{id}',[BookController::class,'update']);
```

contoh implementasinya

```php
<table class="table table-striped table-sm">
    <thead>
        <tr>
            <th scope="col">#</th>
        </tr>
    </thead>
    <tbody>
        @foreach ($books as $book)
            <tr>
                <td>
                    <a href="/book/{{$book->id}}" >
                        <span data-feather="edit"></span> Edit
                    </a>
                </td>
            </tr>
        @endforeach
    </tbody>
</table>
```

Delete

buat method destroy

```php
public function destroy(Book $book)
{
    Book::destroy($book>id);
    return redirect('index');
}
```

buat routenya

```php
Route::delete('/book/{id}',[BookController::class,'destroy']);
```

penggunaanya

PHP

```
<table class="table table-striped table-sm">
    <thead>
        <tr>
            <th scope="col">#</th>
        </tr>
    </thead>
    <tbody>
        @foreach ($books as $book)
            <tr>
                <td>
                    <a href="/book/{{$book->id}}" >
                        <span data-feather="edit"></span> Edit
                    </a>
                    <form action="/book/{{$book->id}}" method="post" class="d-
inline">
                        @method('delete')
                        @csrf
                        <button class="badge bg-danger border-0" onclick="return
confirm('Are You sure?')">
                            <span data-feather="x-circle"></span> Delete
                        </button>
                    </form>
                </td>
            </tr>
        @endforeach
    </tbody>
</table>
```