

PEMOGRAMAN WEB LANJUT

RESUME QUIZ



Oleh :

Akhmad Dian Muzaki

1941720205

PROGRAM STUDI D4 TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
MARET 2021

Routing dan Controller

Semua yang ada hubungannya dengan route telah terdaftar. Konsep tentang route mengikuti pattern nya yaitu Http Verb, Path dan Callback. Pada Http Verb ada get, post, put dan delete masing – masing http ini mewakili satu rule di dalam operasi di Laravel. Setelah ada http verb dilanjutkan dengan path yaitu valid url. Pada callback terdapat dua bentuk : closure dan controller. File Route pada Laravel terdapat routes/web.php, /api.php, /console.php, /channel.php. Pattern routes ditulis “Route::http_verb(‘path’,handler);”.

Kegunaan Http Verb :

Post = create resource

Get = ambil resource

Put = update resource

Delete = Hapus resource

Path (URL)

1. Dengan parameter = ‘mahasiswa/{id}’
2. Tanpa parameter = ‘mahasiswa’

Handler :

1. Closure bentuknya => anonymous function (function yang tidak ada namanya)
Contoh : Route::get(‘mahasiswa’,function(){isinya});
2. Controller => pengganti dari closure
Contoh : Route::get(‘mahasiswa’,[welcome::class,’index’]);

Routes tidak akan bias apa – apa tanpa adanya controller, controller itu seperti otak. Untuk **membuat controller** kita harus mengetik di terminal (**php artisan make:controller WelcomeController**) ini menjadi controller kosong. Ada lagi yang bisa langsung ada isinya atau membuat CRUD yaitu (**php artisan make:controller Welcome --resources**).

```
Route::post('mahasiswa', function ($id) {  
    return "Post Mahasiswa";  
});
```

(Ini dengan Closure) Ketika kita meminta “post” tidak akan bisa nge tes di web browser karena harus mengirim request post dulu.

```
Route::post('dosen', [DosenController::class, 'store']);
```

Routes yang langsung menggunakan controller.

```
$ php artisan route:list
```

Untuk melihat routes apa aja yang telah kita buat.

```
Route::resource('dosen', DosenController::class);
```

Untuk membuat CRUD daripada harus membuat routes satu – satu bisa langsung menggunakan codingan itu.

```
Route::get('mahasiswa', [MahasiswaController::class, 'index']);
```

routes dengan controller.

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class MahasiswaController extends Controller
8 {
9     //
10    public function index()
11    {
12        return "Hello Mahasiswa";
13    }
14 }

```

`Route::get('mahasiswa/{id}',` Routes controller dengan menambahkan parameter{id}. Parameter bisa menerima apa saja ketika kita mengetikkan di browser bagian url.

`te::get('mahasiswa/{id}', MahasiswaController::class, 'detail')` codingan keseluruhannya. Karena kita menambahkan detail maka kita harus membuat method detail pada file Mahasiswa Controller seperti ini

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class MahasiswaController extends Controller
8 {
9     //
10    public function index()
11    {
12        return "Hello Mahasiswa";
13    }
14
15    public function detail($id)
16    {
17        return "Halaman Profile Mahasiswa Dengan Id : " . $id;
18    }
19 }

```

`public function store(Request $request)` Request sangat berguna untuk validasi, ngambil sata dll.

Ketika kita ingin menggunakan request walaupun sudah ada parameter, kita tetap bisa menggunakan tetapi parameter harus diletakkan diakhir seperti ini.

```

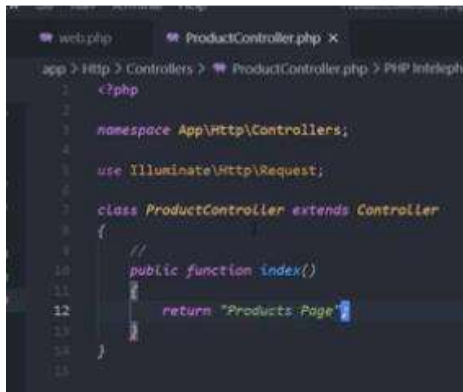
1 public function detail(Request $request, $id)
2 {
3     return "Halaman Profile Mahasiswa Dengan Id : " . $id;
4 }

```

Combo Routes Controller dan View Passing Data dari Controller ke view

1. Buat Routes => jadi di web.php -> kita membuat route dengan controller.
2. Buat Controller => `php artisan make:controller WelcomeController`
3. Konek Controller ke routes => pakai controllernya di routes

Pertama



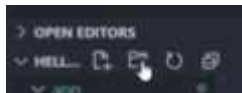
```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ProductController extends Controller
8 {
9     //
10    public function index()
11    {
12        return "Products Page";
13    }
14 }
```

Kedua ini dan akan bisa connect

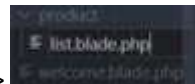


```
12 // Here is where you can register web routes for your application
13 // routes are loaded by the RouteServiceProvider within a group
14 // contains the "web" middleware group. Now create something great
15 //
16
17 //
18
19 Route::get('/', function () {
20     return view('welcome');
21 });
22
23 // /product/{id}
24 // /products/
25
26 Route::get('products', [ProductController::class, 'index']);
```

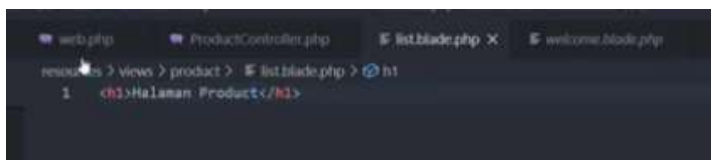
4. Buat view => Buat nya bisa dengan folder baru ataupun tidak



=> missal di dalam product ini akan kita kasih file baru



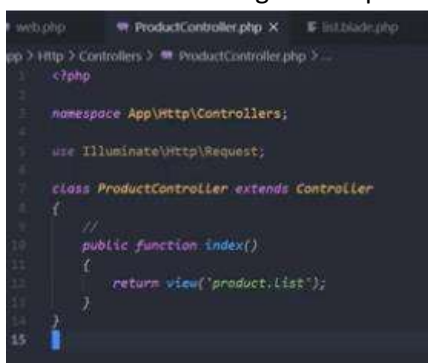
=>



```
1 <h1>Halaman Product</h1>
```

Ketika kita ingin memanggil ini

maka kita harus mengubah di productcontroller menjadi seperti dibawah ini



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ProductController extends Controller
8 {
9     //
10    public function index()
11    {
12        return view('product.list');
13    }
14 }
15
```

5. Konek View ke Controlle => Return View di controller seperti gambar diatas.
6. Kirim data dari Controller ke view =>

Ini adalah data nya. Jika kita ingin kirim data jangan lupa untuk memberikan nama sesuai dengan yang telah kita buat. (Ini menggunakan array)

```
web.php ProductController.php listblade.php welcome.blade.php
app > Http > Controllers > ProductController.php > PHP Intelephense > ProductController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ProductController extends Controller
8 {
9     //
10    public function index()
11    {
12        $category = "Sportswear";
13        $products = ['Sepatu Nike', 'Sepatu Specs', 'Sepatu Swallow'];
14        return view('product.list', ['category' => $category, 'products' => $products]);
15    }
16 }
17
```

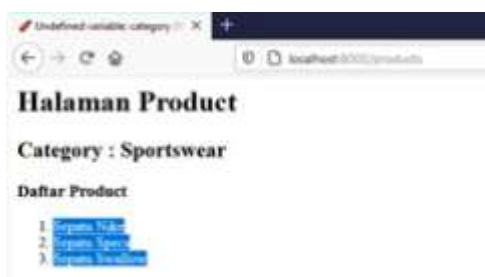
Untuk menampilkan data diatas kita mengetik syntax di list.blade.php yang telah kita buat seperti dibawah ini.

```
resources > views > product > listblade.php > ol
1 <h1>Halaman Product</h1>
2 <h2>Category : {{ $category }}</h2>
3 <h3>Daftar Product</h3>
4
5 <ol>
6     @foreach ($products as $item)
7         <li>{{ $item }}</li>
8     @endforeach
9 </ol>
```

Ketika kita menggunakan ol dan li maka akan menampilkan angka

```
<ol>
    @foreach ($products as $item)
        <li>{{ $item }}</li>
    @endforeach
</ol>
```

Ini adalah hasilnya.



Lanjutan Router Controller View

Ini adalah cara mengirimkan data dari controller ke view, ada 3 cara.

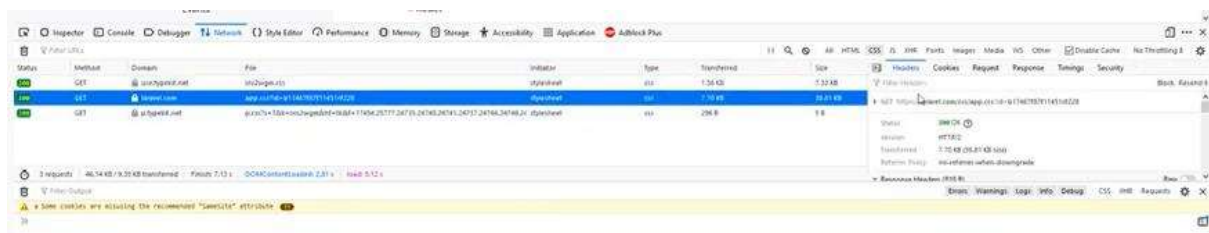
```
web.php • ProductController.php • list.blade.php
app > Http > Controllers > ProductController.php > PHP Intelephense > ProductController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ProductController extends Controller
8 {
9     //
10    public function index()
11    {
12        $category = "Sportswear";
13        $products = ['Sepatu Nike', 'Sepatu Specs', 'Sepatu Swallow'];
14        return view('product.list', ['category' => $category, 'products' => $products]);
15        return view('product.list')->with('category', $category)->with('products', $products);
16        return view('product.list', compact('category', 'products'));
17    }
18 }
19
```

Pengenalan Laravel Mix, nge Build Asset css dan js di laravel

Pada Davelope Web terdapat du acara untuk memasukkan library yaitu CDN (Content Delivery Network) ketika file sudah di compile dan di minifine kita sudah dapat menggunakan akan tetapi kita memerlukan internet. dan Local File.

Laravel Mix adalah package frontend yang disiapkan untuk manajemen asset di Laravel.

Ini adalah contoh hasil dari Laravel mix yang nanti akan meminifine content. Keuntungan meminifine dan mengumpulkan semua css dalam satu file yaitu request nya akan lebih enteng akan tetapi tergantung dengan kecepatan internet dan teknologi browser maka css nya dapat di simpan dalam case.



Bagaimana cara compilenya ?

Dengan mengumpulkan semua file js(JavaScript) dan file css kemudian di compile menjadi satu file.

Pada Laravel mix terdapat dua approve :

1. Approve (DEV) => untuk development, ketika ingin coding di local kita menggunakan ini.
2. Approve (PROD) => untuk production, ketika ingin rilis ke server kita menggunakan ini dan digabung dengan minified.

Cara Menjalankan Laravel Mix

1. Kita mengetikkan => npm install



```
HP@DESKTOP-A5VL0F7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ npm install
[ ] - fetchMetadata: [ ] pacote range nanif
```

2. Ini adalah perintah Meng install bootstrap, library, popper.

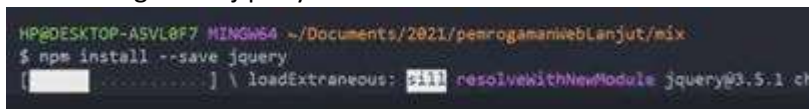


```
HP@DESKTOP-A5VL0F7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ npm install --save bootstrap@next @popperjs/core
[ ] - loadExtraneous: [ ] resolveWithNewModule @pop
```

Bootstrap ada 2 :

- a. App.js => pada js dimasukkan ke dalam js .
- b. App.css => pada css dimasukkan ke dalam css.

3. Cara meng install jquery



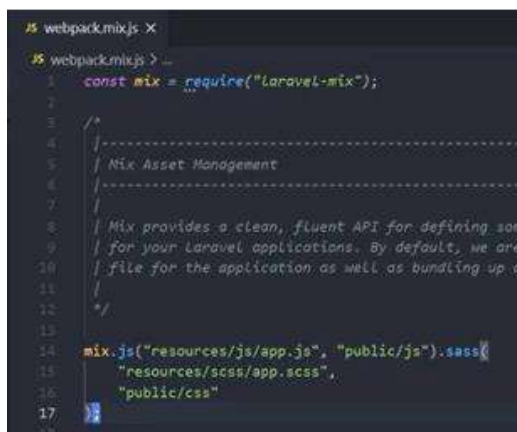
```
HP@DESKTOP-A5VL0F7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ npm install --save jquery
[ ] - loadExtraneous: [ ] resolveWithNewModule jquery@3.5.1 ch
```

Bagaimana caranya bootstrap dan jquery masuk ke file kita :

Kita buka file webpack.mix.js dan hapus pada bagian PostCss hapus sampai “ ; ”. Kemudian tambahkan “.sass” ini kita gunakan untuk meng coding. Sass menggunakan css.



```
1 // webpack.mix.js
2
3 const mix = require('laravel-mix');
4
5 // Mix Asset Management
6
7 // Mix provides a clean, fluent API for defining some asset paths
8 // for your Laravel applications. By default, we are compiling the CSS
9 // file for the application as well as bundling up all the JS files.
10
11
12
13
14 mix.js('resources/js/app.js', 'public/js')
15     .sass('resources/sass/app.scss', 'public/css')
16     .done();
```



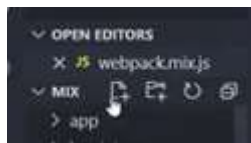
```
1 // webpack.mix.js
2
3 const mix = require('laravel-mix');
4
5 // Mix Asset Management
6
7 // Mix provides a clean, fluent API for defining some asset paths
8 // for your Laravel applications. By default, we are compiling the CSS
9 // file for the application as well as bundling up all the JS files.
10
11
12
13
14 mix.js('resources/js/app.js', 'public/js').sass(
15     'resources/sass/app.scss',
16     'public/css'
17 )
18 .done();
```

Hasilnya.

Jadi kita mengkompilasi file ".css" ke public scss dan menjadi app.scss.

Jika di vscode tidak ada app.scss maka kita dapat membuatnya.

Pertama klik resources => klik tambah folder dan berikan nama scss



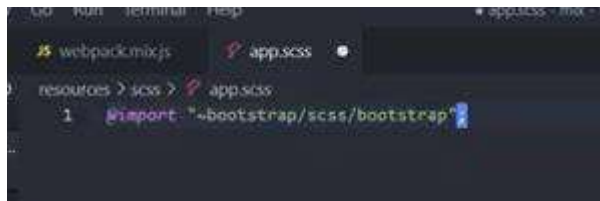
dilanjutkan dengan membuat file



kemudian

kita kasih nama `app.scss`.

Dibawah ini kita bisa menggunakan bootstrapnya



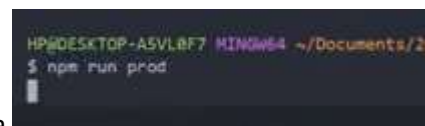
npm run dev disini untuk mendownload app.css dan app.js Laravel mix. setelah ke download kita diminta untuk mengulang sekali lagi walaupun terjadi error pertamanya dibiarkan saja.



File	Size
/js/app.js	597 KiB
css/app.css	189 KiB

Kita dapat mengecilkan ini

dengan cara



Cara memasukkan java scrip bootstrap dan jquery ke dalam bootstrap scss dan app.js



Js, css dan bootstrap kita masukkan lagi dengan mengetiki ini.



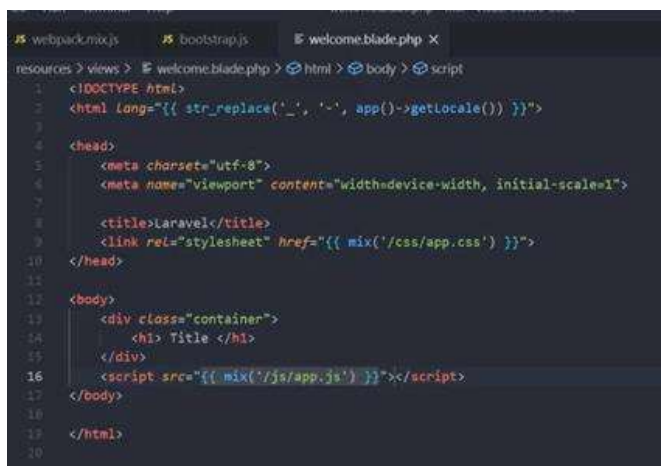
Kita bisa mencobanya dengan membuka file welcome.blade.php kemudian kita hapus (yg di blok) karena sudah tidak dibutuhkan



Kemudian pada bagian body isinya kita hapus



Dan di isikan seperti ini . Kemudian untuk meng import java scrip dan css kita dapat menambah (link href) dan menambah kan lagi import js untuk ke mix nya.



hasilnya seperti ini.

Kita copas codingan



Kita masukkan di tempat div .

```
padmix.js  JS bootstrap.js  welcome.blade.php
> view > welcome.blade.php > html > body > div.container
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">

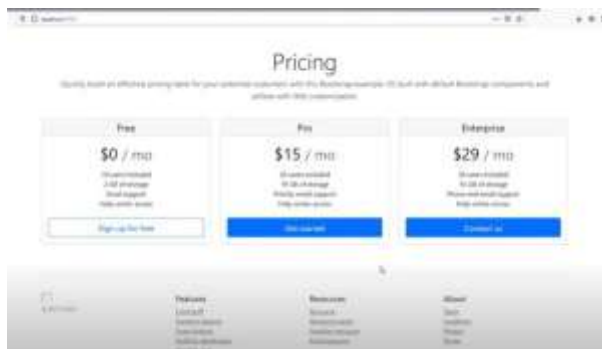
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>Laravel</title>
  <link rel="stylesheet" href="{{ mix('/css/app.css') }}">
</head>

<body>
  <div class="container">
    </div>
  <script src="{{ mix('/js/app.js') }}"></script>
</body>

</html>
```

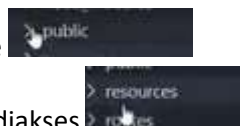
Dan Hasilnya di browser seperti ini



Manajemen Asset Css dan Javascript

Asset Manajemen js, css, dll bisa ditaruh di folder :

1. Public => diakses online
2. Resource => tidak bisa diakses



Kita bisa menggunakan :

1. CDN
2. Local => kalau local kita taruh file di folder public atau bisa menggunakan Laravel mix.

Menggunakan CDN dengan bootstrap



Kita copy link di JsDelivr yang atas kemudian letakkan disini

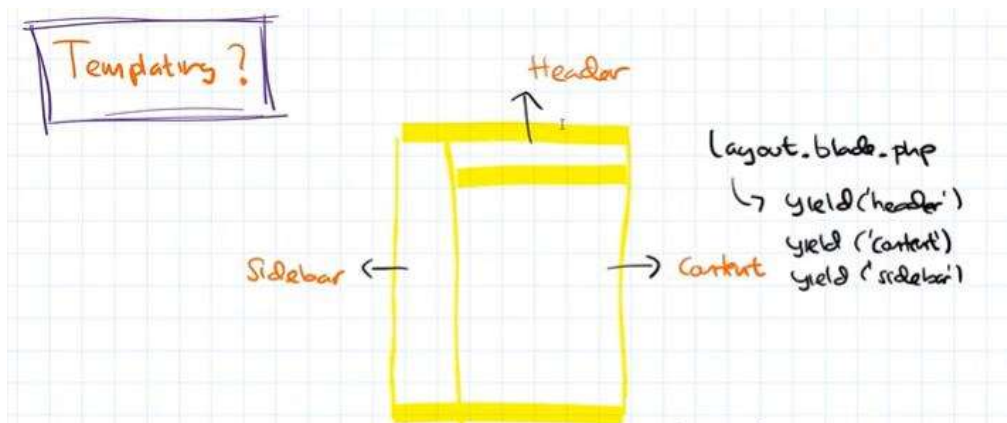
```
resources > views > welcome.blade.php > html > head
1 <!DOCTYPE html>
2 <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7
8   <title>laravel</title>
9   {{-- <link rel="stylesheet" href="{{ mix('/css/app.css') }}" --}}
10  {{-- CSS only --}}
11  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet">
12
13 </head>
14
15 <body>
16   <div class="container">
17     <div class="pricing-header px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
18       <h1 class="display-4">Pricing</h1>
19       <p class="lead">Quickly build an effective pricing table for your potential customers. It's built with default Bootstrap components and utilities with little
20       example. It's built with default Bootstrap components and utilities with little
21     </div>
22
```

Kita copy lagi link di JsDelivr yg bawahnya kemudian letakkan disini.

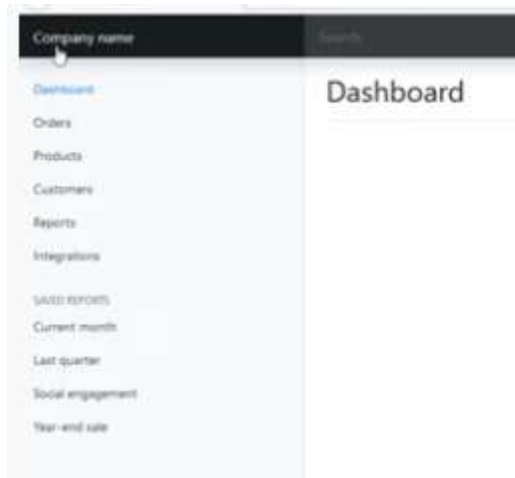
```
resources > views > welcome.blade.php > html > body
106 <ul class="list-unstyled text-sm">
107   <li><a class="link-secondary" href="#">Team</a></li>
108   <li><a class="link-secondary" href="#">Locations</a></li>
109   <li><a class="link-secondary" href="#">Privacy</a></li>
110   <li><a class="link-secondary" href="#">Terms</a></li>
111 </ul>
112 </div>
113 </div>
114 </footer>
115 </main>
116 </div>
117 {{-- <script src="{{ mix('/js/app.js') }}" --}}</script> {{--}}
118 {{-- Javascrpt Bundle with Popper --}}
119 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.js"></script>
120
121 </body>
122
123 </html>
124
```

Ketika menggunakan cdn sever tidak hanya buka ke kita tapi buka juga ke server JsDelivr.

Cara Menggunakan Blade Template Engine



Ketika kita ingin membuat sesuatu yang berbeda dari gambar dibawah ini kita bisa langsung menggunakan (@yield)



Disinilah kita memberikan yield

Fungsi dari **@yield('content')** adalah menampilkan tag html yang berada di dalam **@section('nama_content')**, jadi kita tinggal buat saja **@yield('content')**, **@yield('menu')** atau **@yield('footer')** pada layout. Kemudian isinya buat dalam **@section('menu')/@section('footer')**.

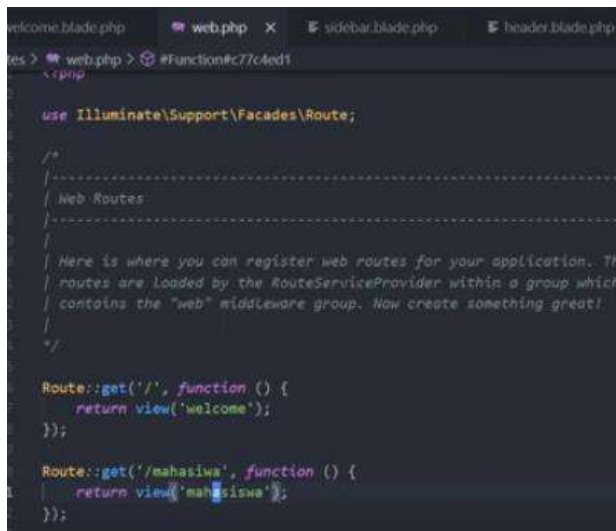
```
sources > views > layoutblade.php > html > body > div.container-fluid > div.row > main.col-md-9.ms-sm-auto.col-lg-10.px-md-4
</style>
</head>
<body>
  <header class="navbar navbar-dark sticky-top bg-dark flex-md-nowrap p-0 shadow">
    @yield('header')
  </header>

  <div class="container-fluid">
    <div class="row">
      <nav id="sidebarMenu" class="col-md-3 col-lg-2 d-md-block bg-light sidebar collapse">
        @yield('sidebar')
      </nav>

      <main class="col-md-9 ms-sm-auto col-lg-10 px-md-4">
        @yield('content')
      </main>
    </div>
  </div>

  <script src="/docs/5.0/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-b5kH1y1q1tYqzWZdNVA9wStHhM1S1LPP1lqV720/TY96U71K0Kf158CwuKc0D381BYkNRz801E00W1gYUSSSF0n30" crossorigin="true">
```

Jika ingin menambah routes baru kita masuk dulu ke web.php



```
welcome.blade.php  web.php  sidebar.blade.php  header.blade.php
resources > web.php > #Function#c77c4ed1
<?php

use Illuminate\Support\Facades\Route;

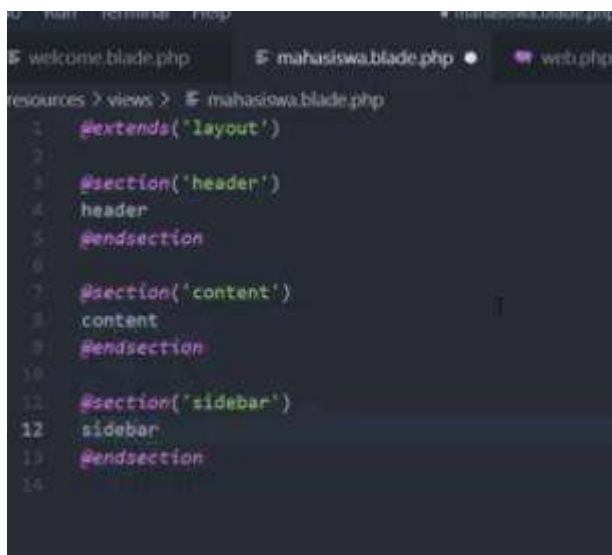
/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. The
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Route::get('/mahasiswa', function () {
    return view('mahasiswa');
});
```

Setelah membuat kita harus membuat file diview dengan nama mahasiswa.

Setiap awalnya harus seperti ini agar dapat mengerjakan.



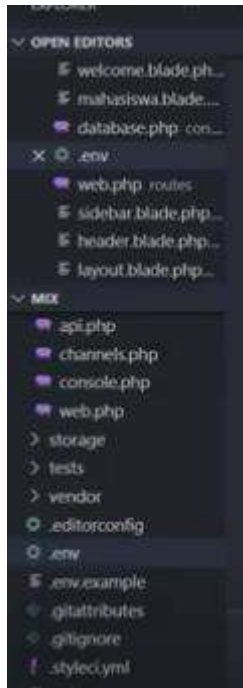
```
resources > views > mahasiswa.blade.php
1  @extends('layout')
2
3  @section('header')
4  header
5  @endsection
6
7  @section('content')
8  content
9  @endsection
10
11 @section('sidebar')
12 sidebar
13 @endsection
14
```



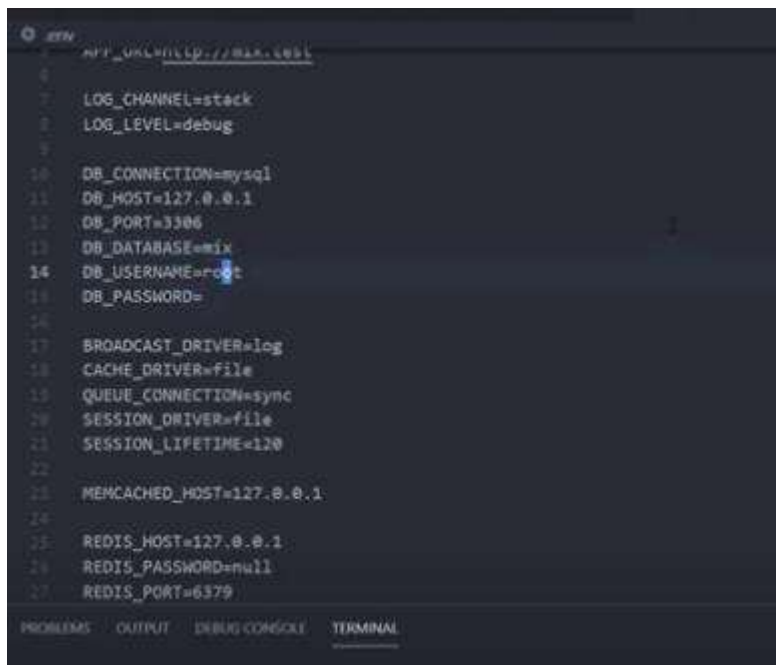
Cara Mengkoneksikan Database Mysql Ke Laravel

Pertama kita harus membuat database dulu dan harus connect.

Cara mestart projectnya ke database. Kita buka (.env)



Yang umum diganti biasanya codingan ke 13,14,15



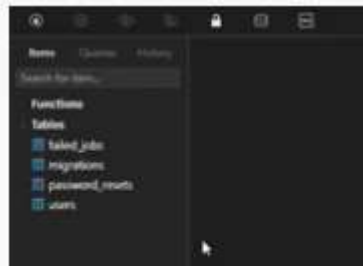
Ketika kita ingin realise ke public codingan ke 14 tidak boleh root dan codingan ke 15 harus ada passwordnya.

Cara mengecek apakah databasenya sudah connect atau tidak. Kita mengetikkan ini



```
HP@DESKTOP-ASVLBF7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ php artisan migrate
```

Kalau sudah connect maka di database nya ada ini. Laravel membawa beberapa tabel yang di masukkan ke migration yang akan kita pakai.



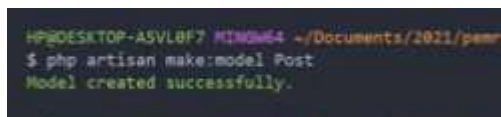
Cara Setup Eloquent Model di Laravel

Artisan Command

-> **Create** Models => Seader => Controller => factory

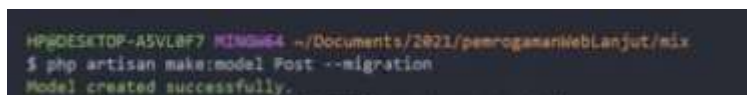


Ketika kita mengetikkan ini maka, akan dibuatkan sebuah file model yang ada di



```
HP@DESKTOP-ASVLBF7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ php artisan make:model Post
Model created successfully.
```

Ketika kita mengetikkan ini akan membuat file migration dengan nama file yang telah kita buat.



```
HP@DESKTOP-ASVLBF7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ php artisan make:model Post --migration
Model created successfully.
```

Ini digunakan untuk membuat contr5oller beserta dengan models



```
HP@DESKTOP-ASVLBF7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ php artisan make:model Post --controller
Model created successfully.
Controller created successfully.
```

Membuat data awal



```
HP@DESKTOP-ASVLBF7 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ php artisan make:model Post --seed
Model created successfully.
Seeder created successfully.
```


Akan membuat file model factory migrate seeder controller

```
HP@DESKTOP-ASVL8F7 MINGW64 ~/Documents/2021/pekerjaanWebLanjut/mix
$ php artisan make:model Post -mfsc
Model created successfully.
Factory created successfully.
Migrating: 2021_03_04_060310_create_posts_table
```

Di Models kita bisa :

- Custom table name
- Custom id
- Custom Increment
- Custom primary key

Ketika kita memiliki tabel post Laravel menganggap kita punya satu tabel yg bernama post.

Tabel ini untuk mengubah tabelnya misal kita punya tabel post dan kita ingin mengganti my post

```
Post.php X
app > Models > Post.php > PHP Intelephense > Post
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Post extends Model
9 {
10     //punya table = posts
11     //id table = id
12     // primary key = auto increment
13     // primary key = integer
14     use HasFactory;
15
16     protected $table = "news";
17     protected $primaryKey = "postId";
18     protected $keyType = "string";
19     protected $incrementing = false;
20 }
21
```

Cara setup Migration

Migration adalah kita bisa Merubah perubahan. Semua perubahan terhadap skema database biasanya akan tercatat di migration.

Ketika kita sudah membuat tabelnya seperti gambar dibawah ini. Tabel nya belum bisa masuk ke

```
HP@DESKTOP-ASVL8F7 MINGW64 ~/Documents/2021/pekerjaanWebLanjut/mix
$ php artisan migrate
Migrating: 2021_03_04_060310_create_posts_table
```

dalam database. Jadi harus mengetik `php artisan migrate` setelah itu tabel akan masuk ke dalam database yang telah kita buat.

```
Post.php 2021_03_04_060310_create_posts_table.php PostController.php
database > migrations > 2021_03_04_060310_create_posts_table.php > PHP Intelephense > CreatePostsTable >
10  * Run the migrations.
11  *
12  * @return void
13  */
14  public function up()
15  {
16      Schema::create('posts', function (Blueprint $table) {
17          $table->id();
18          $table->string("title", 100)->index();
19          $table->string("slug", 100)->index();
20          $table->text("content");
21          $table->boolean("draft")->default(false);
22          $table->timestamps();
23      });
24  }
25
26  /**
27   * Reverse the migrations.
```

Cara Setup Seeder dan Faker

Dengan cara ini dapat membuat migration baru

```
HPDESKTOP-ASVLF77 MINGW64 ~/Documents/2021/pemrogramanWebLanjut/mix
$ php artisan make:migration addImageToPostTable
```

Cara menambahkan tabel dengan temapt yang kita inginkan

```
10  *
11  *
12  * @return void
13  */
14  public function up()
15  {
16      Schema::table('post', function (Blueprint $table) {
17          //
18          $table->string('image')->after('slug')->nullable();
19      });
20  }
21
22  /**
```

Cara drop tabel / menghapus tabelnya

```
10  *
11  *
12  * @return void
13  */
14  public function down()
15  {
16      Schema::table('post', function (Blueprint $table) {
17          //
18          $table->dropColumn('image');
19      });
20  }
21
22  /**
```

Cara membuat Faker

```
HP@DESKTOP-A5VL0F7 MINGW64 ~/Documents/2021/programanWebLanjut/mix
$ composer require fzaninotto/faker
```

Disini kita dapat melihat databsnya

```
HP@DESKTOP-A5VL0F7 MINGW64 ~/Documents/2021/programanWebLanjut/mix
$ php artisan db:seed
Seeding: Database\Seeders\PostSeeder
Seeded: Database\Seeders\PostSeeder (49.95ms)
Database seeding completed successfully.
HP@DESKTOP-A5VL0F7 MINGW64 ~/Documents/2021/programanWebLanjut/mix
```