PRAKTIKUM
PEMROGRAMAN FRAMEWORK

MODUL 3
Routing dan Bundling Asset di Laravel

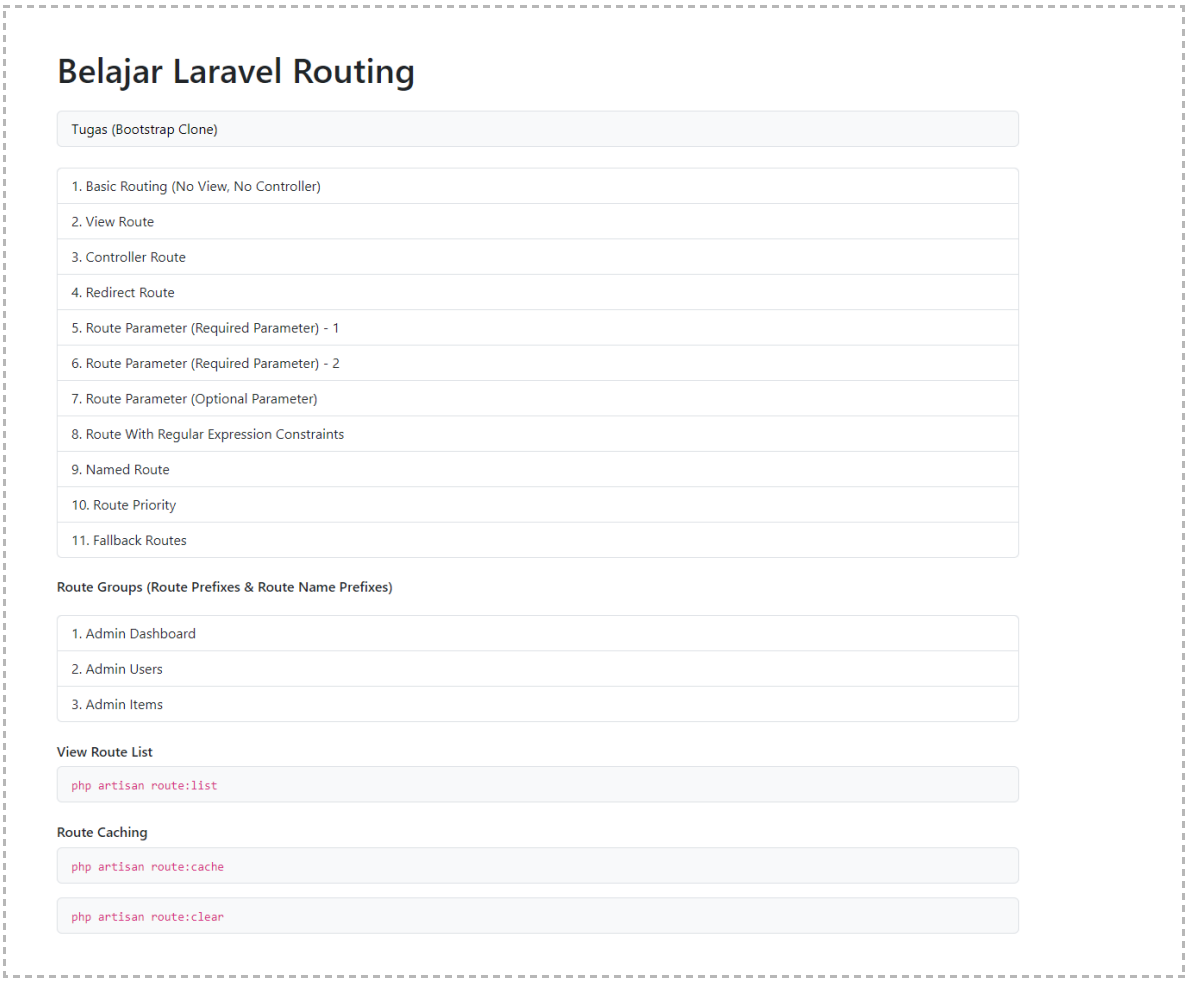
Disusun Oleh:
Purnama Anaking, S.Kom., M.Kom.

PROGRAM STUDI S1 SISTEM INFORMASI
FAKULTAS TEKNOLOGI INFORMASI DAN BISNIS
INSTITUT TEKNOLOGI TELKOM SURABAYA
2023

DAFTAR ISI

MODUL 3
ROUTING DAN BUNDLING ASSET DI LARAVEL

Pada praktikum kali ini kita akan mulai memasuki materi awal pada pembelajaran framework Laravel. Kita akan mulai belajar tentang Routing di Laravel dengan berbagai macam jenisnya. Kemudian kita akan belajar tentang bundling asset pada Laravel. Kegiatan ini dilakukan agar mahasiswa dapat menerapkan konsep routing dan manajemen asset pada framework Laravel.



1. Generate Laravel Project dan Laravel UI

- Buat project laravel baru via composer

```
composer create-project laravel/laravel your-project-name
```

- Masuk ke dalam folder project laravel yang baru saja dibuat, kemudian install package Laravel UI.

```
composer require laravel/ui
```

- Generate scaffolding untuk project Laravel berbasis CSS framework Bootstrap.

```
php artisan ui bootstrap
```

- Jalankan script di bawah ini untuk compile scafolding Bootstrap yang barusan di-setup.

```
npm install
```

- Jalankan script di bawah ini untuk mengaktifkan local development server Laravel sehingga aplikasi web dapat diakses melalui browser.

```
php artisan serve
```

2. Bunding Asset dengan Vite

- Download dan install NodeJS terbaru via **https://nodejs.org/en/download/**
- Periksa hasil instalasi Anda. Ketik script ini di prompt perintah Anda:

```
node -v
```

```
npm -v
```

- Install semua dependencies yang dibutuhkan untuk Bundling Asset dengan Vite, dengan mengetikkan perintah:

```
npm install
```

- Terapkan fitur **"Refreshing on Save"** dengan memeriksa file konfigurasi file vite pada **vite.config.js** dan sesuaikan seperti gambar di bawah ini.

```
import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';

export default defineConfig({
    plugins: [
        laravel({
            input: [
                'resources/sass/app.scss',
                'resources/js/app.js',
            ],
            refresh: true,
        }),
    ],
});
```

- Terapkan fitur **“Processing Static Assets With Vite”** dengan buka file **/resources/js/app.js** lalu sesuaikan kode program seperti di bawah ini. Vite akan merujuk pada path direktori yang kita definiskan untuk mengambil aset gambar/image yang kita butuhkan nantinya.

```
import "./bootstrap";
import.meta.glob(["../images/**"]);
```

- Buat folder pada **/resources** dengan nama **images.** Letakkan aset gambar/image yang akan kita gunakan pada website kita pada folder tersebut.

- Buka file View bernama **welcome.blade.php**. Hapus seluruh kode program yang ada. Kemudian, letakkan dan arahkan file **css** dan **javascript** yang telah didefinisikan di atas, pada file view (blade) menggunakan Blade Directive **@vite()**. Panggil asset gambar/image dengan pendekatan Vite seperti di bawah ini.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Welcome</title>
    @vite('resources/sass/app.scss')
</head>
<body>
    <div class="container m-5">
        {{-- Contoh cara mereferensikan gambar di dalam file blade dengan
menggunakan pendekatan Vite --}}
        <img class="img-thumbnail" src="{{
Vite::asset('resources/images/main.svg') }}" alt="image">
    </div>
    @vite('resources/js/app.js')
</body>
</html>
```

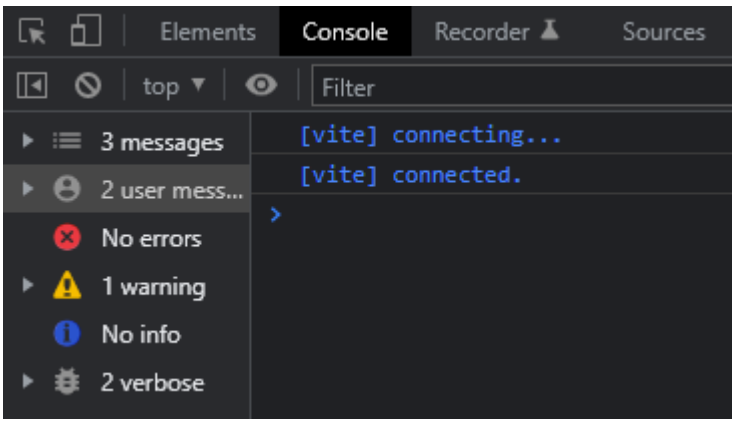
- Jalankan Vite.
    - Untuk mode **development** (jalankan ini saja jika sedang development):

```
npm run dev
```

    - Atau untuk mode **production** (ketika akan deploy ke server):

```
npm run build
```

- Pastikan Vite Connected melalui Tab Console seperti di bawah ini.



- Jika sudah berhasil maka ketika kita mengubah kode program pada file css, blade, ataupun javascript, maka browser akan melakukan **reload secara otomatis**.

3. Install Bootstrap dan Bootstrap Icons Terbaru pada Project Laravel

- Jalankan perintah berikut pada project laravel anda untuk mengintal bootstrap, popper dan bootstrap icons terbaru.

```
npm install bootstrap@5.3.0-alpha3 bootstrap-icons @popperjs/core
```

- Di dalam project laravel anda, buka file **resources\sass\app.scss** dan tambahkan:

```
@import "bootstrap-icons/font/bootstrap-icons.css";
```

- Hapus atau Comment kode import Font & import Variables pada file **resources\sass\app.scss**. Kode program akan terlihat seperti di bawah ini:

```
// Fonts
// @import url("https://fonts.bunny.net/css?family=Nunito");

// Variables
// @import "variables";

// Bootstrap
@import "bootstrap/scss/bootstrap";
@import "bootstrap-icons/font/bootstrap-icons.css";
```

4. Praktik Laravel Routing

- Buka file **routes/web.php,** buat sebuah Route dengan menuliskan kode program seperti di bawah ini.

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

- Buat file View baru pada direktori **/resources/views/** dengan nama **routing.blade.php**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Belajar Laravel Routing</title>
    @vite('resources/sass/app.scss')
</head>
<body>
    <div class="container m-5">
        <h1>Belajar Laravel Routing</h1>

        <div class="list-group list-group-numbered mt-4">
            {{-- Kode anda selanjutnya letakkan di sini --}}
        </div>

        {{-- Khusus kode program untuk Route Groups di sini --}}
    </div>
    @vite('resources/js/app.js')
</body>
</html>
```

- Akses halaman ini dengan mengetikkan **localhost:8000/routing** pada browser.

4.1. Praktik Basic Routing

- Buka file **routes/web.php,** praktekkan **Basic Routing (No View, No Controller)** dengan menuliskan kode program seperti di bawah ini.

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

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/basic_routing') }}" class="list-group-item
list-group-item-action">
    Basic Routing (No View, No Controller)
</a>
```

4.2. Praktik View Route

- Buka file **routes/web.php,** praktekkan **View Route** dengan menuliskan kode program seperti di bawah ini.

```
Route::view('/view_route', 'view_route');
Route::view('/view_route', 'view_route', ['name' => 'Purnama']);
```

- Buat file View dengan nama **view_route.blade.php**, kemudian isikan dengan kode program seperti di bawah ini.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>View Route</title>
    @vite('resources/sass/app.scss')
</head>
<body>
    <div class="container m-5">
        <h1>This is from View Route</h1>
        <p>Hello, My name is {{ $name }}</p>
    </div>
    @vite('resources/js/app.js')
</body>
</html>
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/view_route') }}" class="list-group-item
list-group-item-action">
    View Route
</a>
```

4.3. Praktik Controller Route

- Generate file controller dengan nama RouteController menggunakan bantuan script artisan sebagai berikut: **php artisan make:controller RouteController**
- Tambahkan function bernama **“index”** pada class **RouteController** tersebut.

```
public function index() {
    return "This is from Controller";
}
```

- Buka file **routes/web.php,** praktekkan **Controller Route** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/controller_route', [RouteController::class, 'index']);
```

- Pada file **routes/web.php**, lihat bagian atas file tersebut, pastikan **RouteController** ter-import seperti kode program di bawah ini.

```
<?php

use App\Http\Controllers\RouteController;
use Illuminate\Support\Facades\Route;
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/controller_route') }}" class="list-group-item
list-group-item-action">
    Controller Route
</a>
```

4.4. Praktik Redirect Route

- Buka file **routes/web.php,** praktekkan **Redirect Route** dengan menuliskan kode program seperti di bawah ini.

```
Route::redirect('/', '/routing');
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/') }}" class="list-group-item list-group-item-action">
    Redirect Route
</a>
```

4.5. Praktik Route Parameter (Required Parameter)

- Buka file **routes/web.php,** praktekkan **Route Parameter (Required Parameter)** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/user/{id}', function($id) {
    return "User Id: ".$id;
});
Route::get('/posts/{post}/comments/{comment}', function($postId,
$commentId) {
    return "Post Id: ".$postId.", Comment Id: ".$commentId;
});
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/user/12345') }}" class="list-group-item
list-group-item-action">
    Route Parameter (Required Parameter) - 1
</a>
<a href="{{ url('/posts/01/comments/20') }}" class="list-group-item
```

```
list-group-item-action">
    Route Parameter (Required Parameter) - 2
</a>
```

4.6. Praktik Route Parameter (Optional Parameter)

- Buka file **routes/web.php,** praktekkan **Route Parameter (Optional Parameter)** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('username/{name?}', function($name = null) {
    return 'Username: '.$name;
});
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/username') }}" class="list-group-item
list-group-item-action">
    Route Parameter (Optional Parameter)
</a>
```

4.7. Praktik Route With Regular Expression Constraints

- Buka file **routes/web.php,** praktekkan **Route With Regular Expression Constraints** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/title/{title}', function($title) {
    return "Title: ".$title;
})->where('title', '[A-Za-z]+');
```

- Buka file view **routing.blade.php** lalu tamba hkan kode program sebagai berikut:

```
<a href="{{ url('/title/this-is-my-title') }}" class="list-group-item
list-group-item-action">
    Route With Regular Expression Constraints
</a>
```

- Jika anda test di browser akan muncul halaman warning **404 Not Found**. Analisa dan jelaskan kenapa terjadi demikian!

4.8. Praktik Named Route

- Buka file **routes/web.php,** praktekkan **Named Route** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/profile/{profileId}', [RouteController::class,
'profile'])->name('profileRouteName');
```

- Tambahkan function bernama **“profile”** pada class **RouteController**.

```
public function profile($profileId) {
    return "This is Profile from Controller, profile id: ".$profileId;
}
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ route('profileRouteName', ['profileId' => '123']) }}"
class="list-group-item list-group-item-action">
    Named Route
</a>
```

4.9. Praktik Route Priority

- Buka file **routes/web.php,** praktekkan **Route Priority** dengan menuliskan kode program seperti di bawah ini.

```
Route::get('/route_priority/{rpId}', function($rpId) {
    return "This is Route One";
});
Route::get('/route_priority/user', function() {
    return "This is Route 1";
});
Route::get('/route_priority/user', function() {
    return "This is Route 2";
});
```

- Comment Route yang pertama di atas, seperti kode program di bawah ini

```
// Route::get('/route_priority/{rpId}', function($rpId) {
//     return "This is Route One";
// });
Route::get('/route_priority/user', function() {
    return "This is Route 1";
});
Route::get('/route_priority/user', function() {
    return "This is Route 2";
});
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/route_priority/user') }}" class="list-group-item
list-group-item-action">
    Route Priority
</a>
```

- Test pada browser dengan mengetikkan **localhost:8000/route_priority/user**. Analisa prioritas route yang terjadi sebelum dan sesudah comment Route yang pertama di atas!

4.10. Praktik Fallback Routes

- Buka file **routes/web.php,** praktekkan **Fallback Routes** dengan menuliskan kode program seperti di bawah ini.

```
Route::fallback(function() {
    return 'This is Fallback Route';
});
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<a href="{{ url('/asdqwezxc') }}" class="list-group-item
list-group-item-action">
    Fallback Routes
</a>
```

4.11. Praktik Route Groups (Route Prefixes & Route Name Prefixes)

- Buka file **routes/web.php,** praktekkan **Route Groups (Route Prefixes & Route Name Prefixes)** dengan menuliskan kode program seperti di bawah ini.

```
Route::prefix('admin')->name('admin.')->group(function() {
    Route::get('/dashboard', function() {
        return "This is admin dashboard";
    })->name('dashboard');
    Route::get('/users', function() {
        return "This is user data on admin page";
    })->name('users');
    Route::get('/items', function() {
        return "This is item data on admin page";
    })->name('items');
});
```

- Buka file view **routing.blade.php** lalu tambahkan kode program sebagai berikut:

```
<h6 class="mt-4">Route Groups (Route Prefixes & Route Name Prefixes)</h6>
<div class="list-group list-group-numbered mt-4">
    <a href="{{ route('admin.dashboard') }}" class="list-group-item
list-group-item-action">
        Admin Dashboard
    </a>
    <a href="{{ route('admin.users') }}" class="list-group-item
list-group-item-action">
        Admin Users
    </a>
```

```
    <a href="{{ route('admin.items') }}" class="list-group-item
list-group-item-action">
        Admin Items
    </a>
</div>
```

4.12. Praktik View Route List

- Ketikkan pada cmd / terminal script artisan berikut ini: **php artisan route:list**

4.13. Praktik Route Caching

- Ketikkan pada cmd / terminal script artisan berikut ini untuk menerapkan Route Caching: **php artisan route:cache**
- Ketikkan pada cmd / terminal script artisan berikut ini untuk menghapus Route Cache: **php artisan route:clear**

5. Tugas

1. Praktekkan seluruh poin praktikum yang ada di atas secara **INDIVIDU**.
2. Terapkan cloning website bootstrap yang telah dikerjakan pada modul 1 praktikum minggu sebelumnya ke dalam project laravel yang anda buat di praktikum ini.
    a. Tambahkan Route dan file View baru agar halaman Bootstrap Clone ini bisa diakses dari browser.
    b. Jika ada kode css yang perlu ditambahkan, bisa ditambahkan pada file **resources/sass/app.scss**
3. Dokumentasikan hasil praktikum dan soal no.2 tersebut **(screenshot kode program, output pada browser, penjelasan kode program yang ditulis)** dalam bentuk Laporan Praktikum.
4. **Upload manual** project laravel hasil praktikum ke github (hapus folder **node_modules** & **vendor**).
    a. Buat **repository public** baru pada akun github anda.
    b. Klik link **“Uploading an existing file”** untuk meng-upload project laravel yang telah anda buat.
    c. Cantumkan URL repository yang telah dibuat pada Laporan Praktikum.
5. Kumpulkan Laporan Praktikum **(.pdf)** yang telah dibuat di dalam praktikum + tugas praktikum (no. 2) via **E-Learning** paling lambat sebelum jadwal praktikum minggu depan.