LAPORAN MODUL 5

CONTROLLER

Dosen Pengampu: LINTANG SETYO KURNIAWATI S.Kom., M.Pd

Disusun Oleh:

Bayu Krisna (2010651095)

PROGRAM STUDI TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH JEMBER

2022

CONTROLLER

Controller merupakan bagian dari konsep MVC yang bertugas untuk memproses semua request untuk ditampilkan kembali melalui view. Bisa dikatakan controller merupakan jembatan penghubung antara model dan view. Jika sebelumnya kita masih ada mengatur proses pada route, kini tidak saatnya lagi karena itu merupakan tugas dari controller. Pada aplikasi yang akan dibuat nanti semua route diarahkan ke controller dan tidak ada lagi proses seperti mengambil data dari database atau memanggil method view pada route. Pada modul sebelumnya kita telah menggunakan satu class controller yaitu produkContrller, namun pada modul ini kita akan lebih mendalami cara penggunaan controller.

Untuk membuat Controller, kita tidak perlu membuatnya secara manual, karena Laravel memiliki fitur artisan yang sudah dibahas pada bagian sebelumnya untuk mempermudah pembuatan Controller. Silahkan jalankan terminal dan arahkan ke projek laravel yang dibuat kemudian tuliskan script berikut (masih dengan projek Laravel penjualan): *php artisan make:controller pelangganController*

Perintah tersebut akan menghasilkan file controller dengan nama **pelangganController.php** pada direktori **penjualan/app/Http/Controller**. Untuk contoh penggunaan controller pada proses Laravel pertama siapkan sebuah route pada file **web.php** yang ada di direktori **routes/web.php** seperti pada gambar dibawah:

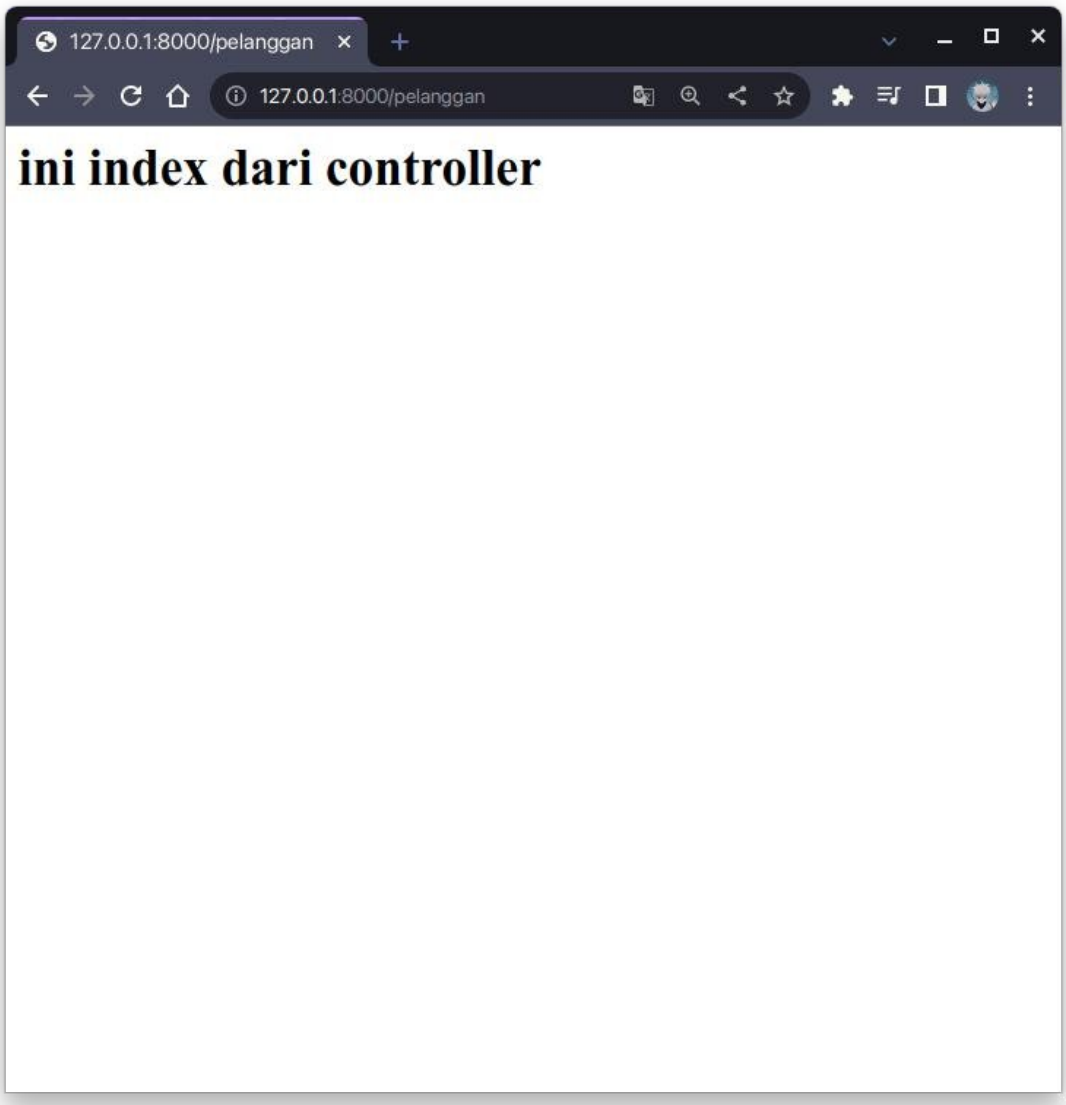
```php
Route::get('/pelanggan', [pelangganController::class, 'index']);
```

Kemudian pada file **pelangganController.php** yang ada pada direktori **app\Http\controller** yang baru saja dibuat, buatlah satu function atau method yang bernama **index()** dan isikan script seperti berikut:

Untuk mengetes apakah pembuatan fungsi pada controller sudah berhasil maka

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class pelangganController extends Controller
{
    public function index()
    {
        echo '<h1>ini index dari controller</h1>';
    }
}

```

bukalah alamat route yang baru saja dibuat yaitu **http://127.0.0.1:8000/pelanggan** (pastikan server artisan telah aktif) maka hasilnya akan nampak seperti berikut:



Atau pada funtion **index()** yang ada pada **pelangganController.php** kita dapat langsung melakukan return view atau mengarahkan route yang diakses ke suatu halaman, ubahlah function **index()** yang ada pada **pelangganController** lalu buatlah menjadi seperti berikut:

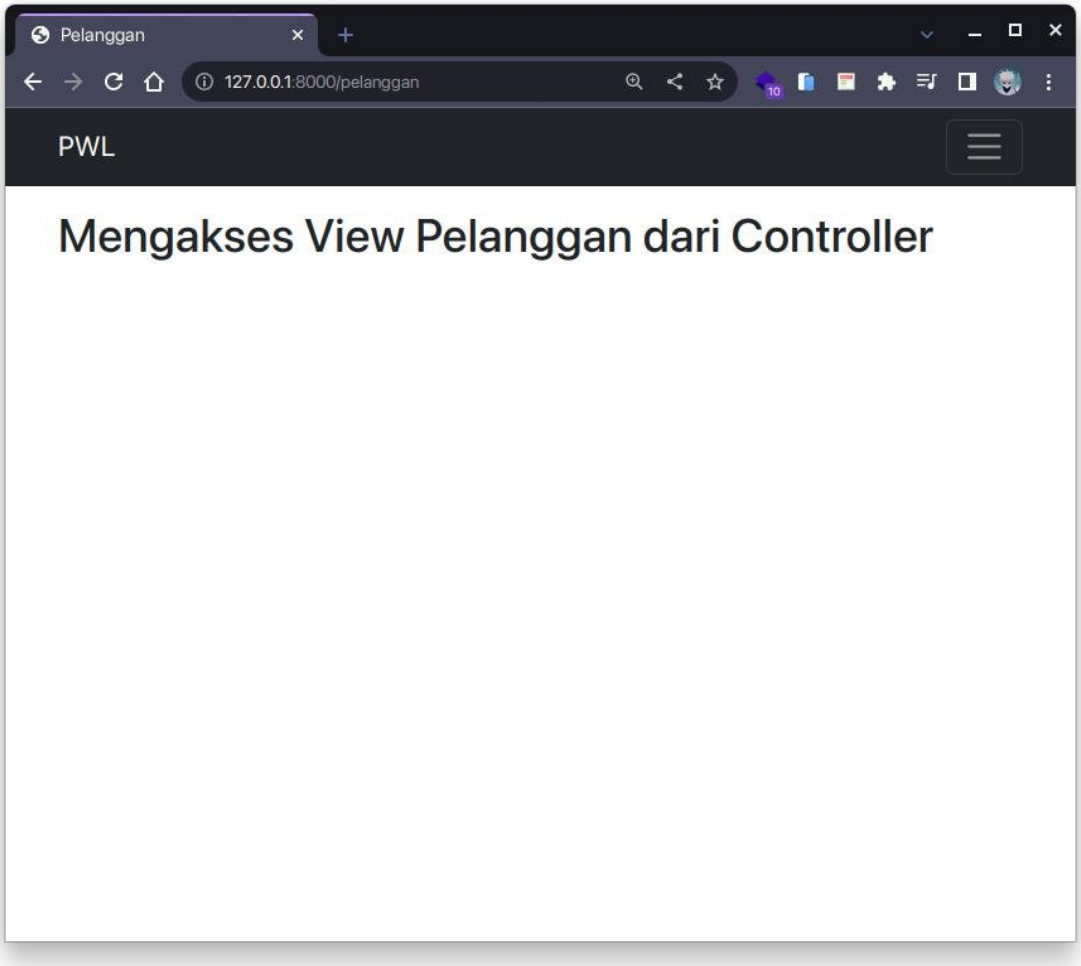
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class pelangganController extends Controller
{
    public function index()
    {
        return view('pelanggan/index', [
            'title' => 'Pelanggan'
        ]);
    }
}
```

Kemudian buatlah satu direktori dalam direktori **resources/view** bernama **pelanggan** dan buat satu file yang bernama **index.blade.php** dan isikan script seperti berikut:

Kemudian akses kembali route **http://127.0.0.1:8000/pelanggan** atau reload halaman

```php
@extends('layouts.main')

@section('container')
    <h1>Mengakses View Pelanggan dari Controller</h1>
@endsection
```

yang telah dibuka sebelumnya, maka tampilan nya akan menjadi seperti berikut:



Pada gambar diatas adalah halaman view yang diakses dari route **http://127.0.0.1:8000/pelanggan** yang route tersebut mengarahkan kita ke **pelangganController.php** dengan function yang dituju adalah function **index()**. Di dalam function index yang ada di dalam **pelangganController.php** halaman diarahkan dengan menggunakan method view ke dalam view **index.blade.php** yang ada pada direktori **resourcess/views/pelanggan**.

MENGAKSES FUNGSI BERBEDA DALAM SATU CONTROLLER

Di dalam konsep laravel kita dapat memanggil suatu fungsi dari fungsi lain di dalam laravel menggunakan method **this**. Method this ini menandakan kita akan memanggil fungsi yang ada di dlam satu controller yang sama dengan bentuk pemanggilan adalah untuk fungsi yang tidak melakukan return value:

Sedangkan untuk fungsi yang melakukan return value kita perlu mendefinisikan suatu variabel sebelum memanggil fungsi tersebut untuk tempat menyimapan return value dari fungsi yang diakses. Untuk contoh pemanggilan fungsi yang memiliki return value dapat dilihat pada contoh dibawah:

```php
$data = $this->namaFungsi();
```

Untuk contoh penggunaan dari pemanggilan fungsi dalam satu controller yang sama kita masih akan menggunakan route **http://127.0.0.1:8000/pelanggan** namun pada **pelangganController.php** tambahkan satu fungsi bernama **dataPelanggan()** dan isikan script seperti dibawah:

Pada fungsi **dataPelanggan()** diatas kita mendefinisikan satu variabel array

```php
public function dataPelanggan()
    {
        $pelanggan = ['Ina','Ani','Ita','Indra'];
        return $pelanggan;
    }
```

bernama pelanggan dan langsung melakukan return value variabel pelanggan tersebut, kemudian pada fungsi **index()** yang pada controller yang sama yaitu **pelangganController.php** kita modifikasi sedikit scriptnya menjadi seperti berikut:

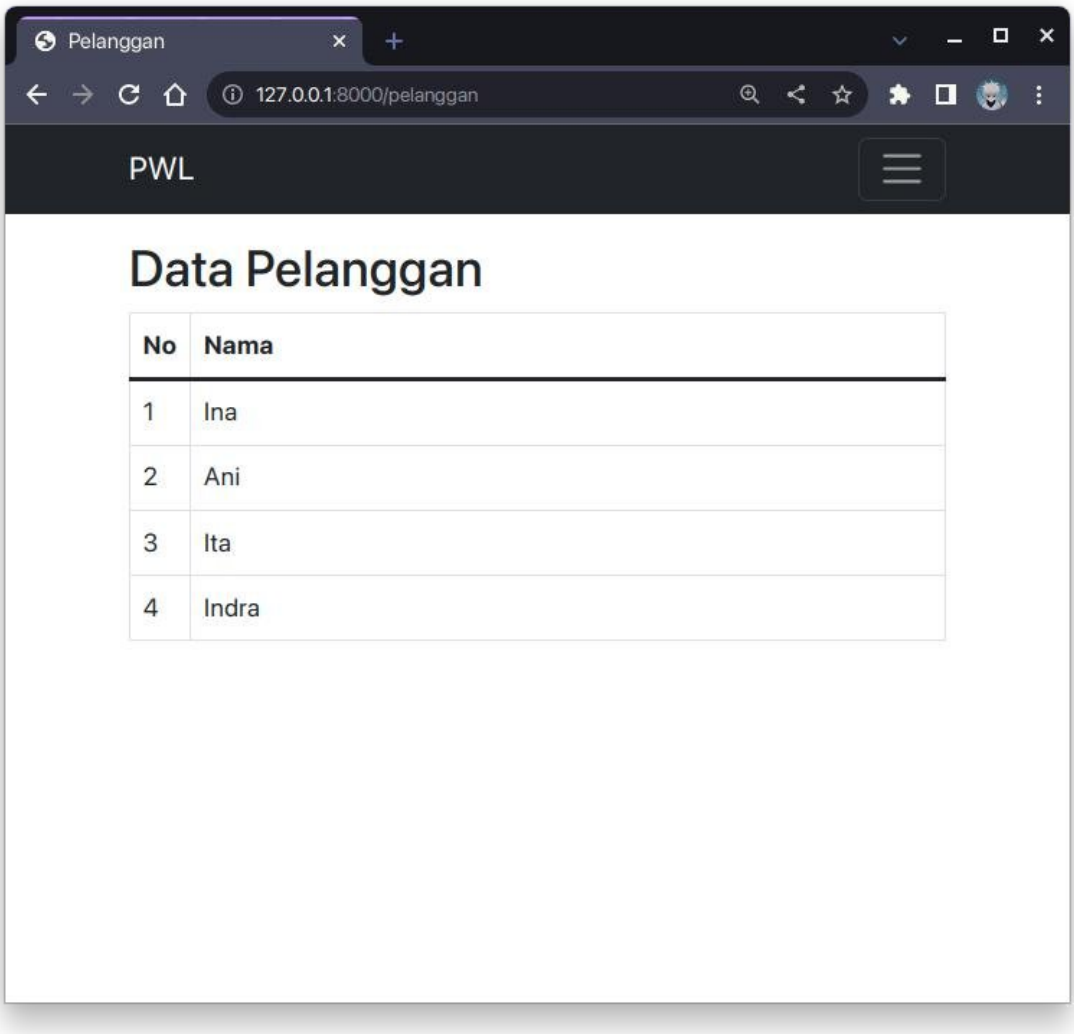
```php
return view('pelanggan/index', [
            'title' => 'Pelanggan',
            'pelanggan' => $this->dataPelanggan()
        ]);
```

Pada fungsi **index()** diatas kita memanggil fungsi **dataPelanggan()** menggunakan method **this** dan menerima nilai dari return value yang kita tampung kedalam variabel pelanggan yang ada di dalam fungsi **index()**. Untuk menampilkan data pelanggan yang telah kita terima dari fungsi **dataPelanggan()**, modifikasi view **index.blade.php** yang berada pada direktori **resources/views/pelanggan** menjadi seperti dibwah:

Kemudian jalankan kembali route **http://127.0.0.1:8000/pelanggan** maka hasilnya

```
@extends('layouts.main')

@section('container')
    <h1>Data Pelanggan</h1>

    <div class="row">
      <div class="col-md-4">
        <table class="table table-bordered" style="width: 100%">
          <thead>
            <tr>
              <th style="width: 7%">No</th>
              <th>Nama</th>
            </tr>
          </thead>
          <tbody class="table-group-divider">
            @foreach ($pelanggan as $i => $cust)
            <tr>
              <td style="width: 7%">{{ $i+1 }}</td>
              <td>{{ $cust }}</td>
            </tr>
            @endforeach
          </tbody>
        </table>
      </div>
    </div>
@endsection
```

akan nampak seperti pada gambar dibawah:



Untuk menghasilkan controller yang lengkap, buatlah satu controller baru dengan nama produkController tapi tambahkan kata **--resource** dibelakangnya sehingga menjadi seperti berikut: *php artisan make:controller produkController –resource*

perintah tersebut akan menghasilkan file produkController yang lengkap dengan 7 fungsi yang dibutuhkan untuk melakukan proses CRUD pada laravel seperti berikut:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class produkController extends Controller
{

    public function index()
    {
        //
    }

    public function create()
    {
        //
    }

    public function store(Request $request)
    {
        //
    }

    public function show($id)
    {
        //
    }

    public function edit($id)
    {
        //
    }

```

```
35      public function update(Request $request, $id)
36      {
37          //
38      }
39
40      public function destroy($id)
41      {
42          //
43      }
44  }
45
```

Adapun kegunaan dari 7 fungsi tersebut adalah sebagai berikut:

1. **index()** merupakan fungsi utama controller yang akan dipanggil ketika action tidak disebutkan. Bisanya digunakan untuk menaruh script untuk menampilkan data.

2. **create()** merupakan fungsi untuk menampilkan form tambah data.
3. **store()** merupakan fungsi untuk menaruh script untuk menyimpan data yang dikirim dari form tambah data untuk disimpan ke database.

4. **show()** merupakan funsgi untuk menaruh script yang menampilkan data lebih detail dari sebuah record.

5. **edit()** merupakan fungsi untuk menaruh script untuk mengarahkan ke form edit.
6. **update()** fungsi ini digunakan untuk menauh script yang akan digunakan untuk memperbaharui data dari database yang diterima dari form edit.

7. **destroy()** untuk menaruh script yang digunakan untuk menghapus data dari database.

Kelebihan lain yang di dapat dari pembuatan controller dengan **--resource** adalah kita cukup membuat satu route yang mengarah ke controller tanpa harus membuat route masing-masing fungsi. Untuk membuktikannya tambah route berikut ke file **web.php** yang ada di direktori **routes**.

```
1  Route::resource('/produk', produkController::class);
```

Dengan mendefinisikan satu route di atas, Laravel akan membuatkan route untuk setiap fungsi pada controller. Untuk membuktikannya, silahkan ketikan script artisan berikut pada terminal dan jangan lupa arahkan ke direktori project laravel yang dibuat.

Contoh penggunaan controlller, misalnya pada fungsi **index()** dapat kita isi dengan script sebagai berikut:

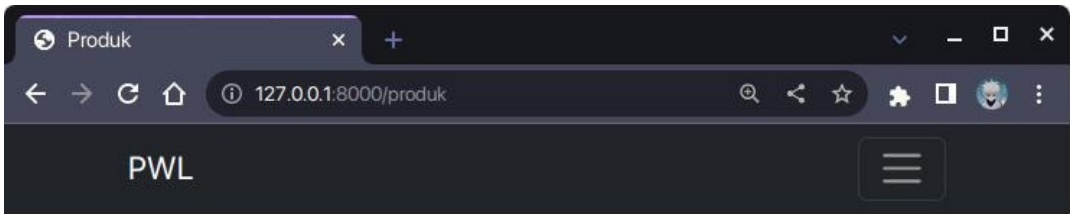
Kemudian buatlah satu view baru di direktori **resourcess/views/produk** (jika

```php
public function index()
    {
        return view('produk/index', [
            'title' => 'Produk',
            'produk' => ['Meja', 'Kursi', 'Buku', 'Lampu']
        ]);
    }
```

direktori produk tidak ada silahkan dibuat) yang bernama **index.blade.php** dan isikan script sebagai berikut:

```
@extends('layouts.main')

@section('container')
    <h1>Data Produk</h1>

    <div class="row">
      <div class="col-md-4">
        <table class="table table-bordered" style="width: 100%">
          <thead>
            <tr>
              <th style="width: 7%">No</th>
              <th>Nama</th>
            </tr>
          </thead>
          <tbody class="table-group-divider">
            @foreach ($produk as $i ⇒ $prod)
            <tr>
              <td style="width: 7%">{{ $i+1 }}</td>
              <td>{{ $prod }}</td>
            </tr>
            @endforeach
          </tbody>
        </table>
      </div>
    </div>
@endsection
```

Kemudian silahkan buka **http://127.0.0.1:8000/produk**, maka hasilnya akan nampak seperti berikut:



Data Produk

No	Nama
	Meja
2	Kursi
3	Buku
4	Lampu