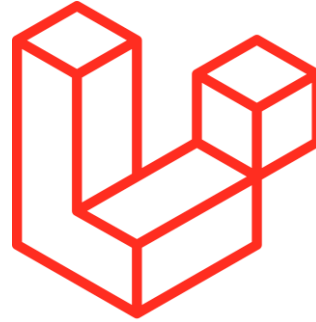


Pemrograman Backend



Pesantren PeTIK II YBM PLN

Jl. KH. Bisri Syansuri RT/01 RW/05, Plosogeneng,
Kec. Jombang, Kabupaten Jombang, Jawa Timur



6. Koneksi Database dan Eloquent ORM



Koneksi Database

- Setelah kita belajar untuk membuat template Laravel framework dengan RWD Bootstrap maka langkah selanjutnya adalah saatnya kita melakukan interaksi ke dalam database.
- Yakni dengan input dari user aplikasi, kita bisa membuat data baru, mengupdate atau menghapus data yang sudah ada.
- Cara termudah dan yang direkomendasikan sesuai dengan arsitekur MVC di Laravel adalah menggunakan Model dan memanfaatkan Eloquent.



Konfigurasi Koneksi Database

Bagi Anda untuk yang menggunakan XAMPP silahkan buka file **.env** yang berada di root folder aplikasi Anda. Lalu isikan seperti kode di dalam slide dengan penjelasan kode program sebagai berikut:

```
DB_CONNECTION=mysql //koneksi databasenya ke database mysql
DB_HOST=localhost //hostnya berada di lokal
DB_PORT=3306 //nomor port database mysql
DB_DATABASE=dblaravel //nama basis data
DB_USERNAME=root //user basis data
DB_PASSWORD=rahasia //password user basis data
```





Metode Bekerja dengan Database



Metode Code First

Dengan metode ini, kita tidak perlu mendesain, membuat database bahkan melakukan insert data awal/dummy melalui aplikasi yang biasa kita gunakan seperti phpMyAdmin, Adminer dan lain-lain. Karena kita cukup mendesain dan membuat desain database langsung dari kode kita. Untuk membuat skema database kita memanfaatkan fitur Laravel yaitu Migration, sedangkan untuk mengisi data awal atau data dummy kita bisa menggunakan Seeding.



Metode Database First

Metode kedua adalah database first, hal ini jika kita sudah memiliki database, tabel dan skemanya. Kita langsung ingin menggunakannya tanpa membuat database baru. Metode ini juga bisa diterapkan jika kita lebih suka mendesain skema tabel langsung dari phpMyAdmin misalnya. Dengan metode ini, kita tidak perlu menggunakan Migration.



Migration

Migration berfungsi sebagai version control database kita. Dengan migration kita bisa membuat, mengubah atau menghapus struktur tabel dan field database tanpa harus membuka aplikasi GUI database management. Dengan Migration inilah, kita bekerja dengan database menggunakan metode Code First.





Membuat File Migration

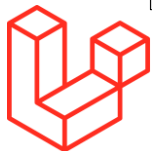
Pertama kita perlu membuat sebuah file Migration, file ini nantinya bisa dieksekusi agar dapat mengubah database sesuai perintah yang kita tuliskan di file tersebut. Untuk membuat file Migration kita gunakan perintah di bawah ini:

```
Command Prompt
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\nasrul>e:

E:\>cd xampp\htdocs\example-app

E:\xampp\htdocs\example-app>php artisan make:migration create_pegawai_table
Created Migration: 2021_06_17_074538_create_pegawai_table
```





Isi File Migration(1)

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreatePegawaiTable extends Migration
{
    public function up()
    {
        Schema::create('pegawai', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('pegawai');
    }
}
```



Isi File Migration(2)

File Migration ini memiliki 2 method yaitu:

- Method `up()` digunakan untuk menuliskan perintah pembuatan atau perubahan struktur database.
- Method `down()` digunakan untuk menuliskan kode yang membatalkan apa yang telah dieksekusi di method `up()`.

Pada method `up()`, pertama kita buat sebuah tabel dengan nama pegawai, hal itu dilakukan melalui static method dari facade Schema, yaitu `Schema::create("pegawai")`. Setelah itu kita definisikan field apa saja yang akan dibuat, secara default perintah: `make:migration` akan membuat 2 field untuk kita yaitu `id` yang merupakan key dan `autoIncrement`, kemudian timestamps yang akan membuat field `created_at` dan `updated_at` pada tabel pegawai.





Membuat Kolom

Setelah membuat file migration tabel pegawai, langkah selanjutnya adalah membuat field/kolom pada tabel pegawai seperti kode di bawah ini:

```
public function up(){  
    Schema::create('pegawai', function (Blueprint $table) {  
        $table->increments('id');  
        $table->integer('nip');  
        $table->string('nama',50);  
        $table->text('alamat');  
        $table->string('email')->unique();  
        $table->timestamps();  
    });
```



Eksekusi Migration

Setelah membuat kolom-kolom pada tabel pegawai, Langkah selanjutnya adalah mengeksekusi migration. Buka terminal dan masuk ke root path aplikasi Laravel kita, jalankan perintah di bawah ini:

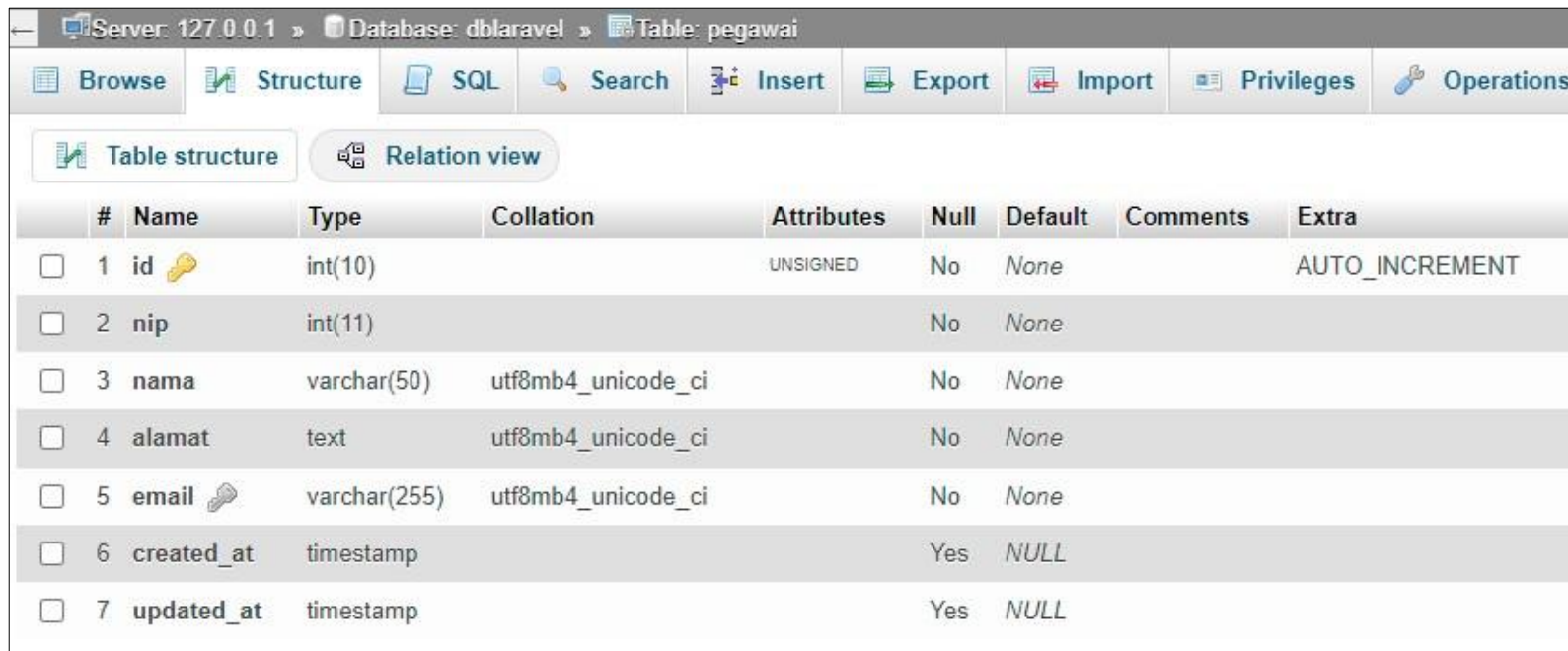
```
E:\xampp\htdocs\example-app>php artisan migrate  
Migrating: 2021_06_17_074538_create_pegawai_table  
Migrated: 2021_06_17_074538_create_pegawai_table (552.89ms)
```

Dengan perintah di atas, maka file-file Migration yang berisi perintah pengubahan database akan dijalankan, hanya kode di method up() yang dieksekusi. Untuk melihat perubahannya, kita bisa melihat langsung tabel di database kita. Coba jalankan perintah migrate di atas setelah kita berhasil menggenerate file migration.



Hasil Eksekusi Migration



Setelah berhasil menjalankan perintah migration, coba buka PHPMyadmin untuk mengecek apakah benar tabel sudah berhasil dibuat.



Server: 127.0.0.1 » Database: dblaravel » Table: pegawai

Browse Structure SQL Search Insert Export Import Privileges Operations

Table structure Relation view

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	id 	int(10)		UNSIGNED	No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	nip	int(11)			No	None		
<input type="checkbox"/>	3	nama	varchar(50)	utf8mb4_unicode_ci		No	None		
<input type="checkbox"/>	4	alamat	text	utf8mb4_unicode_ci		No	None		
<input type="checkbox"/>	5	email 	varchar(255)	utf8mb4_unicode_ci		No	None		
<input type="checkbox"/>	6	created_at	timestamp			Yes	NULL		
<input type="checkbox"/>	7	updated_at	timestamp			Yes	NULL		

Sedding

- Seeding pada laravel adalah sebuah fitur untuk mengisi data pada database dengan data sembarang atau data testing.
- Secara pengertian seed dalam bahasa indonesia berarti benih. Maka sebagaimana benih, seeder dapat digunakan untuk membuat sample data atau dummy data dengan command yang sederhana.
- Maka anda tidak perlu repot untuk melakukan penginputan data secara berulang pada saat proses testing.
- Hal ini tentunya akan mempercepat proses development yang anda lakukan. Mengapa? Karena anda cukup sekali membuat “benih data” yang dapat digunakan secara berulang kali saat dibutuhkan.

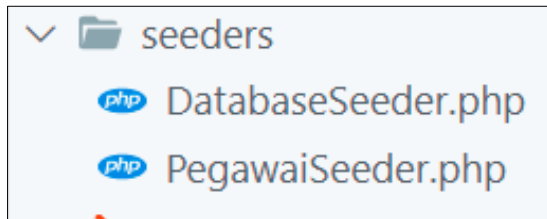


Membuat Sedding

Untuk membuat sedding dari tabel pegawai yang sudah ada, caranya adalah buka terminal dan masuk ke root path aplikasi Laravel kita, jalankan perintah di bawah ini:

```
E:\xampp\htdocs\example-app>php artisan make:seeder PegawaiSeeder  
Seeder created successfully.
```

Setelah perintah di atas berhasil dieksekusi, lihatlah di project Anda di folder seeders ada file PegawaiSeeder.php, seperti tampak gambar di bawah ini:



Mengisi Data Sedding

Untuk mengisi/mengenerate data di tabel pegawai, caranya adalah buka file PegawaiSeeder.php yang berada di folder seeders, lalu isi data seperti kode program di bawah ini:

```
use DB; //tambahkan kode ini di atasclass                                extends Seeder
PegawaiSeeder
//di dalam fungsi run isi data-data pegawai
public function run(){
    for ($i=0; $i < 10; $i++) {
        DB::table('pegawai')->insert(
            [
                'nip' => rand(),
                'nama' => uniqid('nama_'),
                'alamat' => uniqid('alamat_'),
                'email' => uniqid(). '@gmail.com',
                'created_at' => new \DateTime,
                'updated_at' => null,
            ]
        );
    }
}
```



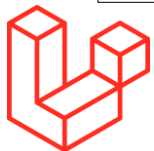
Eksekusi Data Seeding

Setelah mengisi data pegawai sesuai dengan kolom-kolomnya, langkah selanjutnya adalah mengeksekusi data seedingnya. Buka terminal dan masuk ke root path aplikasi Laravel kita, jalankan perintah di bawah ini:

```
E:\xampp\htdocs\example-app>php artisan db:seed --class=PegawaiSeeder  
Database seeding completed successfully.
```

Dengan perintah di atas, maka tabel pegawai akan terisi data pegawai yang baru. Anda bisa cek di phpMyAdmin.

id	nim	nama	alamat	email	created_at	updated_at
1	111	Fawwaz	Bambu Apus	fawwaz@gmail.com	2021-06-17 09:26:10	NULL



Eloquent

- ORM (Object Relational Mapping) bisa melakukan operasi CRUD ke database tanpa harus menulis SQL Command.
- ORM inilah yang bertugas untuk menulis SQL Command untuk kita.
- Laravel menggunakan ORM yang bernama Eloquent.
- Eloquent ORM pada Laravel menyediakan implementasi Active Record yang berarti bahwa setiap model yang kita buat dalam struktur MVC kamu sesuai dengan tabel dalam database kita.



Keunggulan Eloquent(1)

- 1. Sintaks Lebih Ringkas**
- 2. Mudah Mengganti Database**

Dengan Eloquent ORM apabila kita ingin mengganti database dari MySQL ke MSSQL atau Oracle atau yang lain kita tidak perlu khawatir perbedaan sintaks yang bisa membuat error. Karena kita tidak perlu mengubah kode yang kita tulis dengan Eloquent ORM. Eloquent ORM tersebut yang akan generate SQL Command sesuai dengan database baru yang kita pakai.



Keunggulan Eloquent(2)

3. Mudah mengelola relationship antar tabel

Dengan Eloquent ORM membuat, mendapatkan dan memanipulasi relationship antar table menjadi lebih menyenangkan. Jika sebelumnya kita memerlukan SQL Command yang cukup panjang dan rawan kesalahan.

4. Memudahkan pemula dalam SQL Command

Jika Kamu tergolong pemula dalam SQL Command, kamu akan sangat terbantu dengan adanya Eloquent ORM. Karena kamu tidak perlu menghafal SQL Command yang rumit untuk bisa membuat fitur canggih.



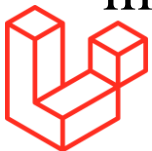
Keunggulan Eloquent(2)

3. Mudah mengelola relationship antar tabel

Dengan Eloquent ORM membuat, mendapatkan dan memanipulasi relationship antar table menjadi lebih menyenangkan. Jika sebelumnya kita memerlukan SQL Command yang cukup panjang dan rawan kesalahan.

4. Memudahkan pemula dalam SQL Command

Jika Kamu tergolong pemula dalam SQL Command, kamu akan sangat terbantu dengan adanya Eloquent ORM. Karena kamu tidak perlu menghafal SQL Command yang rumit untuk bisa membuat fitur canggih.





Contoh Penggunaan Eloquent

```
<?php
// Menampilkan semua data dalam tabel pegawai
Pegawai::all()

// Menampilkan data dalam tabel pegawai yang
sesuai
// dengan parameter
Pegawai::find($id)

// Menghapus data pegawai berdasarkan IDnya
Pegawai::delete($id)
?>
```



Query Builder

- Query Builder pada Laravel menyediakan antarmuka untuk membuat dan menjalankan query dalam database.
- Query Builder menggunakan PDO parameter binding untuk untuk melindungi aplikasi kita dari serangan injeksi SQL jadi kita tidak perlu lagi melakukan filter string sebagai binding.





Contoh Penggunaan Query Builder

```
<?php
```

```
// Menampilkan semua data dalam tabel pegawai
```

```
DB::table('pegawai')->get();
```

```
// Menampilkan data dalam tabel pegawai yang sesuai  
// dengan parameter
```

```
DB::table('pegawai')->where('id',$id)->first();
```

```
// Menghapus data pegawai berdasarkan IDnya
```

```
DB::table('pegawai')->where('id',$id)->delete();
```

```
?>
```



Menampilkan Data dari Database

Membuat Model

Membuat Controller

Membuat Fungsi di Controller

Membuat View

Menambahkan Route

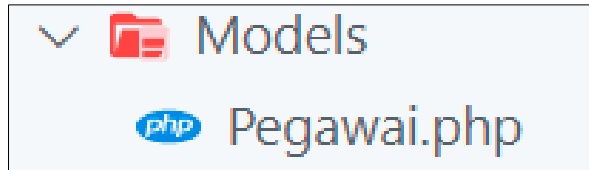


Membuat Model

Untuk membuat model dari tabel pegawai yang sudah tersedia, buka terminal dan masuk ke root path aplikasi Laravel kita, jalankan perintah di bawah ini:

```
E:\xampp\htdocs\example-app>php artisan make:model Pegawai  
Model created successfully.
```

Dengan perintah di atas, maka di dalam folder models akan ada file baru Pegawai.php.



Konfigurasi Model

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Pegawai extends Model
{
    use HasFactory;
    //tambahkan kode untuk mapping ke tabel pegawai
    protected $table = 'pegawai';
}
```



Membuat Controller

Langkah selanjutnya adalah membuat controller, buka terminal dan masuk ke root path aplikasi Laravel kita, jalankan perintah di bawah ini:

```
E:\xampp\htdocs\example-app>php artisan make:controller PegawaiController  
Controller created successfully.
```

Dengan perintah di atas, maka di dalam folder app/Http/controller akan ada file baru PegawaiController.php.



Membuat Fungsi di Controller

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
//tambahan kode dibawah ini
use DB;
use App\Models\Pegawai;
class PegawaiController extends Controller
{
    public function index() //fungsi untuk menampilkan data pegawai
    {
        //dapatkan seluruh data pegawai dengan query builder
        $ar_pegawai = DB::table('pegawai')->get();
        //arahkan ke halaman baru dengan menyertakan data pegawai(compact)
        //di resources/views/pegawai/index.blade.php
        return view('pegawai.index',compact('ar_pegawai'));
    }
}
```





Membuat View

```
resources > views > pegawai > index.blade.php > ...
1  @extends('adminlte::page')
2  @section('title', 'Data Pegawai')
3  @section('content_header')
4      <h1>Data Pegawai</h1>
5  @stop
6  @section('content') {{-- Isi Konten Data Pegawai --}}
7  @php
8      $ar_judul = ['No', 'NIP', 'Nama', 'Alamat', 'Email'];
9      $no = 1;
10 @endphp
11 > <table class="table table-striped">...
30 </table>
31 @stop
32 @section('css')
33     <link rel="stylesheet" href="/css/admin_custom.css">
34 @stop
35 @section('js')
36     <script> console.log('Hi!'); </script>
37 @stop
```





Isi Konten Pegawai(1)

```
@extends('adminlte::page')
@section('title', 'Data Pegawai')
@section('content_header')
    <h1>Data Pegawai</h1>
@stop
@section('content') {{-- Isi Konten Data Pegawai --}}
@php
$ar_judul = ['No', 'NIP', 'Nama', 'Alamat', 'Email'];
$no = 1;
@endphp
<table class="table table-striped">
    <thead>
        <tr>
            @foreach($ar_judul as $jdl)
                <th>{{ $jdl }}</th>
            @endforeach
        </tr>
    </thead>
```





Isi Konten Pegawai(2)

```
{{-- Lanjutan Isi Konten Data Pegawai --}}
```

```
<tbody>
  @foreach($ar_pegawai as $peg)
    <tr>
      <td>{{ $no++ }}</td>
      <td>{{ $peg->nip }}</td>
      <td>{{ $peg->nama }}</td>
      <td>{{ $peg->alamat }}</td>
      <td>{{ $peg->email }}</td>
    </tr>
  @endforeach
</tbody>
</table>
@stop
```



Menambahkan Routing Baru

Selanjutnya tambahkanlah routing baru untuk menampilkan data pegawai. Silahkan buka file **routes/web.php** lalu tambahkan kode dibawah ini:

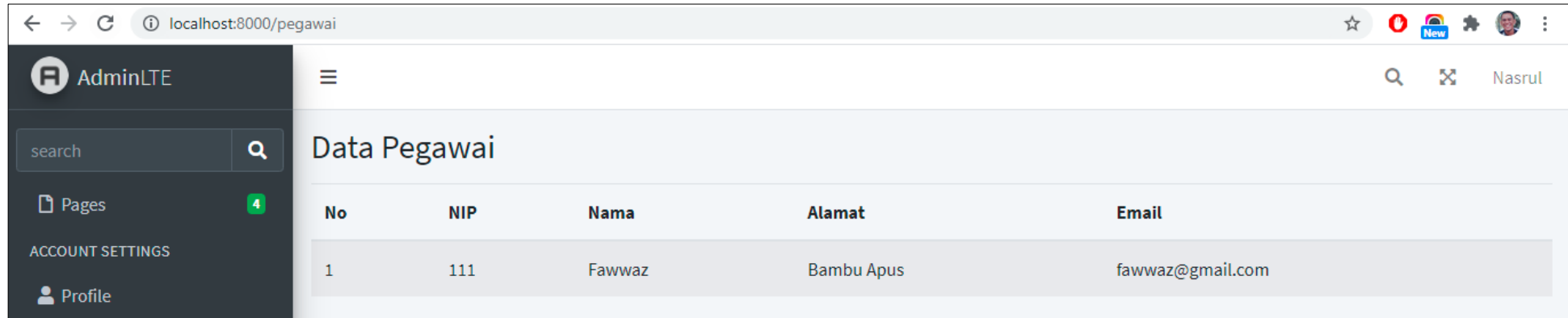
```
//Tambahkan route baru di routes/web.php
use App\Http\Controllers\PegawaiController;

Route::get(
    '/pegawai',
    [PegawaiController::class, 'index']
);
```



Menampilkan Data Pegawai

Selanjutnya dari CMD di project kita jalankan perintah: `php artisan serve`. Maka dalam route di url bisa diakses `http://localhost:8000/pegawai`, maka tampilah seperti di bawah ini:



No	NIP	Nama	Alamat	Email
1	111	Fawwaz	Bambu Apus	fawwaz@gmail.com



**TERIMA KASIH
ATAS SEGALA PERHATIAN
SEMOGA BERMANFAAT...**

