**Catatan laravel**

Perintah artisan laravel

**1.check connection database to app**

php artisan tinker -> DB::connection()->getPdo();

**2. composer dump-autoload (**memperbarui file autoloader)

**-**Gunakan ketika membuat file baru di laravel misalnya setelah membuat helper

**3.membuat migration**

**Tips Penggunaan Migration:­­**

* Gunakan nama migration yang deskriptif, misalnya create\_users\_table atau add\_address\_to\_users\_table.
* Selalu buat backup database sebelum melakukan perubahan pada struktur tabel.
* Gunakan fitur foreign untuk membuat relasi antar tabel.

-membuat file migration (php artisan make:migration nama\_migration)

**Contoh struktur table migration yang bagus**

*public* function up(): void

    {

        Schema::create('ms\_user', function (Blueprint $table) {

            $table->bigIncrements('id'); *// ID utama dengan tipe big integer*

            $table->string('username', 50)->unique(); *// Username unik dengan panjang maksimum 50 karakter*

            $table->string('password'); *// Password (hashed)*

            $table->string('email', 100)->unique(); *// Email unik dengan panjang maksimum 100 karakter*

            $table->string('fullname', 100); *// Nama lengkap pengguna*

            $table->boolean('is\_active')->default(true); *// Status aktif pengguna, default true (aktif)*

            $table->timestamp('last\_login')->nullable(); *// Waktu login terakhir*

            $table->timestamps(); *// Kolom `created\_at` dan `updated\_at`*

            $table->softDeletes(); *// Kolom `deleted\_at` untuk soft delete*

        });

    }

*/\*\**

*\* Reverse the migrations.*

*\*/*

*public* function down(): void

    {

        Schema::dropIfExists('ms\_user');

    }

**-menjalankan migrate**

1.php artisan migrate (akan menjalankan semua migrate yang ada)

2.php artisan migrate --path=/database/migrations/2024\_09\_23\_024515\_tb\_products.php (akan menjalankan migrate secara spesifik file yg di tuju )

**-menghapus table migration di db**

1.php artisan migrate:rollback

**-Mengubah Struktur Tabel**

Jika Anda ingin menambahkan atau mengubah kolom pada tabel yang sudah ada, buat migration baru dengan perintah:

**php artisan make:migration add\_column\_to\_products\_table --table=products**

**Contoh struktur table migration**

**public function up()**

**{**

**Schema::table('products', function (Blueprint $table) {**

**$table->integer('stock')->after('price'); // Menambahkan kolom stock setelah kolom price**

**});**

**}**

**public function down()**

**{**

**Schema::table('products', function (Blueprint $table) {**

**$table->dropColumn('stock'); // Menghapus kolom stock**

**});**

**}**

Setelah edit file jalankan php artisan migrate

**-mengubah nama field di migration**

**php artisan make:migration rename\_field\_in\_table\_name --table= products**

**public function up()**

**{**

**Schema::table('ms\_user', function (Blueprint $table) {**

**$table->renameColumn('old\_name', 'new\_name');**

**});**

**}**

**/\*\***

**\* Reverse the migrations.**

**\***

**\* @return void**

**\*/**

**public function down()**

**{**

**Schema::table('ms\_user', function (Blueprint $table) {**

**$table->renameColumn('new\_name', 'old\_name');**

**});**

**}**

Setelah edit file jalankan php artisan migrate

**Pastikan Menggunakan Package doctrine/dbal**

Metode renameColumn memerlukan paket doctrine/dbal agar dapat berfungsi. Jika paket ini belum terpasang, Anda perlu menginstalnya dengan perintah berikut:

**composer require doctrine/dbal**

**Jika Anda ingin mengembalikan semua migration ke kondisi awal, gunakan perintah berikut:**

**php artisan migrate:reset**

**Untuk mengulang semua migration, gunakan perintah berikut:**

**php artisan migrate:refresh**

**4.composer install (Kapan Menggunakan composer install di Laravel?)**

 **Saat Meng-clone atau Mengunduh Proyek Laravel Baru:** Agar semua dependensi diinstal sesuai dengan composer.lock.

 **Setelah Menghapus Folder vendor:** Untuk menginstal ulang semua dependensi.

 **Sebelum Deploy ke Server Produksi:** Menginstal dependensi tanpa mengubah versi paket, terutama untuk memastikan stabilitas aplikasi di server produksi.

**5.composer update**

**6.Sintaks-sintaks yang sering digunakan**

**1.Try Throwable**

Try catch Throwable(try yang cocok digunakan saat insert dan update) try ini akan menangkap semua jenis kesalahan code ataupun db

**try {**

**//code...**

**} catch (\Throwable $th) {**

**//throw $th;**

**}**

**2.Try Exception**

Try catch exception) try ini akan menangkap semua jenis kesalahan code ataupun db secara spesifik

**try {**

**//code...**

**} catch (\Exception $e) {**

**//throw $e;**

**}**

**3.return redirect route with**

**Untuk mengembalikan pesan dan redirect ketika suatu process selesai**

**return redirect()->route('index.item')->with('success','success save');**

**return redirect()->route('index.item')->with(Error,Failed save : '.** **$e->getMessage());**

**4.Construct**

**Inisialisasi code agar berjalan dengan konsep clean**

**protected $ItemModel;**

**public function \_\_construct(ItemModel $ItemModel) {**

**$this->ItemModel = $ItemModel;**

**}**

**4.Crypt**

Untuk menginkripsi suatu code atau id di laravel

Cara menggunkan Crypt

**use Illuminate\Support\Facades\Crypt;**

panggil Crypt(untuk mengenkripsi id)

**Crypt::encrypt($row->id)**

panggil dcrypt(untuk uncripsi id)

**Crypt::decrypt($id);**

**5.** **find()**

Mencari data dengan id spesifik

**use App\Models\User;**

**$user = User::find(1);**

**if ($user) {**

**// Jika user ditemukan**

**echo $user->name;**

**} else {**

**// Jika user tidak ditemukan**

**echo "User tidak ditemukan";**

**}**

**6.** **findOrFail**

Jika Anda ingin memastikan bahwa data ditemukan dan ingin mendapatkan data secara langsung tanpa perlu mengecek null, Anda bisa menggunakan findOrFail():

**use App\Models\User;**

**$user = User::findOrFail(1);**

**// Jika user ditemukan, akan dikembalikan model**

**// Jika user tidak ditemukan, akan melemparkan exception ModelNotFoundException**

**echo $user->name;**

**7.** **Menemukan Model dengan Kondisi Tertentu (get)**

**use App\Models\User;**

**$users = User::where('id', '>', 5)->get();**

**foreach ($users as $user) {**

**echo $user->name . "<br>";**

**}**

**8.all()**

**­­mengambil semua data**

**$getalldata = $this->UserModel->all();**

**9.validate**

//in controller

**$validasi = $request->validate([**

**‘name’ => ‘required’,**

**])**

//send to db

**$item = $validasi[‘name’];**

//in view

**@error(name)**

**<div class="text-danger">{{ $message }}</div>**

**@enderror**