Ở các bài trước mình đã giới thiệu với mọi người cách truy xuất dữ liệu trong Laravel Với [Query Buider](http://toidicode.com/query-builder-trong-laravel-11.html) và [Eloquent ORM](http://toidicode.com/eloquent-orm-trong-laravel-13.html) rồi. Hôm nay tiếp tục với series [Học Laravel](http://toidicode.com/laravel-14) của Toidicode.com mình xin được giới thiệu với mọi người một thuật ngữ mới nữa trong Laravel đó là **Migratios,**

**1, Migrations trong Laravel là gì?**

-Migration trong Laravel  giống như một control database có tác dụng quản lý cũng như lưu trữ lại cấu trúc của database giúp cho việc sửa đổi database trở lên dễ dàng hơn.

-Điều kiện để sử dụng Migations:

* Phải có kết nối với database .
* Migrations muốn sử dụng được thì phải nằm trong thư mục **App\database\migrations**

**2, Tạo migrations.**

-Để Tạo Migrations thì các bạn cũng có 2 cách tạo là dùng tay và dùng lệnh, nhưng mình khuyến khích mọi người dùng lệnh.

-Tạo Migrations bằng lệnh thì các bạn mở cmd lên và trỏ tới thư mục chứa project của các bạn(như mọi khi :-) ) và gõ 1 trong các lệnh sau tùy theo mục đích của bạn.

* **php artisan make:migration TenMigrate**  : Tạo migrations thông thường.
* **php artisan make:migration TenMigrate --create=TableName**  : Tạo migrations cho bảng.
* **php artisan make:migration TenMigrate --table=TableName**  : Tạo migrations chỉnh sửa bảng.

-Chú Thích: **TenMigrate,TableName** là các thông số các bạn có thể tùy chỉnh.

VD: Mình sẽ tạo Migrations create\_users\_table cho table users.

php artisan make:migration create\_users\_table --create=users

-Nếu tạo thành công nó sẽ báo dạng như sau: Created migration: xxxxxxxxxxxx: Lúc này bạn có thể kiểm tra lại bằng cách truy cập vào  **App\database\migrations**nếu thấy  có file tên trùng với phần xxxxxxxx ở trên thì là đã thành công.

-Tiếp đó các bạn mở file ra và sẽ thấy nội dung có dạng:

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::create('users', function (Blueprint $table) {

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

//

}

}

**Hàm up và hàm down.**

-Hàm up trong Migrations có tác dụng thực thi migration

-Hàm down trong Migrations có tác dụng thực thi đoạn lệnh rollback(trở về trước đó).

**3, Các cú pháp trong Migrations.**

-Dưới đây là một số các câu lệnh tạo bảng hay dùng trong Migrations.

| **Lệnh** | **Chức năng** |
| --- | --- |
| $table->bigIncrements('id'); | Tạo cột id khóa chính tự động tăng kiểu bigint |
| $table->bigInteger('votes'); | Tạo cột votes với kiểu bigint |
| $table->binary('data'); | Tạo cột data với kiểu blob |
| $table->boolean('confirmed'); | Tạo cột confirmed với kiểu boolean |
| $table->char('name', 4); | Tạo cột name với kiểu char tối đa 4 kí tự |
| $table->date('created\_at'); | Tạo cột created\_atvới kiểu date |
| $table->dateTime('created\_at'); | Tạo cột created\_atvới kiểu dateTime |
| $table->dateTimeTz('created\_at'); | Tạo cột name với kiểu DATETIME (with timezone) |
| $table->decimal('amount', 5, 2); | Tạo cột name với kiểu DECIMAL |
| $table->double('column', 15, 8); | Tạo cột name với kiểu DOUBLE |
| $table->enum('choices', ['foo', 'bar']); | ENUM equivalent for the database. |
| $table->float('amount', 8, 2); | FLOAT equivalent for the database, 8 digits in total and 2 after the decimal point. |
| $table->increments('id'); | Incrementing ID (primary key) using a "UNSIGNED INTEGER" equivalent. |
| $table->integer('votes'); | INTEGER equivalent for the database. |
| $table->ipAddress('visitor'); | IP address equivalent for the database. |
| $table->json('options'); | JSON equivalent for the database. |
| $table->jsonb('options'); | JSONB equivalent for the database. |
| $table->longText('description'); | LONGTEXT equivalent for the database. |
| $table->macAddress('device'); | MAC address equivalent for the database. |
| $table->mediumIncrements('id'); | Incrementing ID (primary key) using a "UNSIGNED MEDIUM INTEGER" equivalent. |
| $table->mediumInteger('numbers'); | MEDIUMINT equivalent for the database. |
| $table->mediumText('description'); | MEDIUMTEXT equivalent for the database. |
| $table->morphs('taggable'); | Adds unsigned INTEGER taggable\_id and STRING taggable\_type. |
| $table->nullableTimestamps(); | Same as timestamps(). |
| $table->rememberToken(); | Adds remember\_token as VARCHAR(100) NULL. |
| $table->smallIncrements('id'); | Incrementing ID (primary key) using a "UNSIGNED SMALL INTEGER" equivalent. |
| $table->smallInteger('votes'); | SMALLINT equivalent for the database. |
| $table->softDeletes(); | Adds nullable deleted\_at column for soft deletes. |
| $table->string('email'); | VARCHAR equivalent column. |
| $table->string('name', 100); | VARCHAR equivalent with a length. |
| $table->text('description'); | TEXT equivalent for the database. |
| $table->time('sunrise'); | TIME equivalent for the database. |
| $table->timeTz('sunrise'); | TIME (with timezone) equivalent for the database. |
| $table->tinyInteger('numbers'); | TINYINT equivalent for the database. |
| $table->timestamp('added\_on'); | TIMESTAMP equivalent for the database. |
| $table->timestampTz('added\_on'); | TIMESTAMP (with timezone) equivalent for the database. |
| $table->timestamps(); | Adds nullable created\_at and updated\_at columns. |
| $table->timestampsTz(); | Adds nullable created\_at and updated\_at (with timezone) columns. |
| $table->unsignedBigInteger('votes'); | Unsigned BIGINT equivalent for the database. |
| $table->unsignedInteger('votes'); | Unsigned INT equivalent for the database. |
| $table->unsignedMediumInteger('votes'); | Unsigned MEDIUMINT equivalent for the database. |
| $table->unsignedSmallInteger('votes'); | Unsigned SMALLINT equivalent for the database. |
| $table->unsignedTinyInteger('votes'); | Unsigned TINYINT equivalent for the database. |
| $table->uuid('id'); | UUID equivalent for the database. |

-Nhiều quá các bạn chịu khó Google translate nhé : :-P

**-VD: Mình sẽ tạo bảng users ở trên với các cột:**

* id; khóa chính tự động tăng.
* name: kiểu dữ liệu varchar.
* email:kiểu dữ liệu varchar và là duy nhất(unique).
* password: kiểu dữ liệu varchar.

Code:

public function up()

{

Schema::create('users', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('name');

$table->string('email')->unique();

$table->string('password');

});

}

-còn nếu các bnj muốn khi rollback thì sẽ xóa bảng users thì các bạn viết đoạn code sau:

public function down()

{

Schema::dropIfExists('users');

}

**4, Thực thi Migrations.**

-Sau khi đã tạo và viết code cho migrate thì tất nhiên là chúng ta cần phải chạy nó đúng không nhỉ?(không thì viết làm chi)

-Các lệnh thực thi migrations:

|  |  |
| --- | --- |
| php artisan migrate | chạy migration |
| php artisan migrate:resest | resest lại migration |
| php artisan migrate:refesh | chạy lại migration |
| php artisan migrate:status | xem trạng thái của migration |
| php artisan migrate:install | cài đặt migration |

VD: Với bảng users ở trên mình sẽ tiến hành chạy lệnh php artisan migrate Kết quả:

