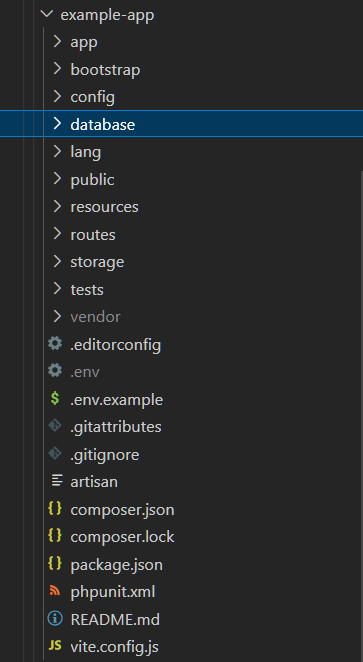
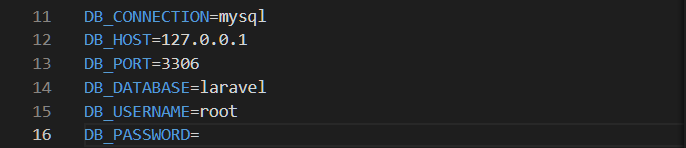
# 資料遷移 (Data Migration)

遷移是一種資料庫的版本控制，讓團隊能夠輕鬆的修改跟共享應用程式的資料庫結構。遷移通常會搭配 Laravel 的結構建構器，讓你可以輕鬆的建構應用程式的資料庫結構。

## 如何進行資料遷移?

1. 打開.env查看資料庫的基本設定



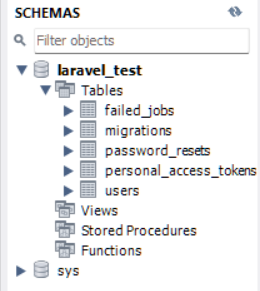


這是.env中的初始設定

1. 開啟mysql的連結

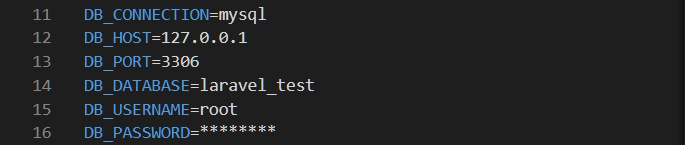
可以透過Xampp或是MySQL Workbench或是其他開啟資料庫的工具，獲取進入資料庫的連結方式。

此處使用的是MySQL Workbench開始資料庫的連結。



並且建立一個新的資料庫，以做後續的連動/資料遷移的目標 => 此處為laravel\_test

1. .env中設定MySQL的環境



回到.env中設定先前新增的資料庫名稱作為資料遷移的目標

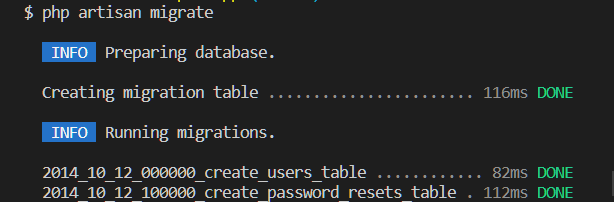
新增連結進MySQL WorkBench的密碼

做好上述的工作，即可完成資料遷移的前置作業

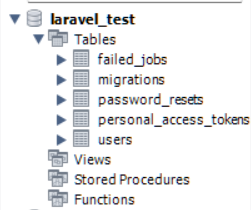
1. 資料遷移

透過git bash的terminal，輸入遷移資料的程式碼

php artisan migrate



資料遷移的動作即刻進行



遷移完畢後，會出現上述的幾個表

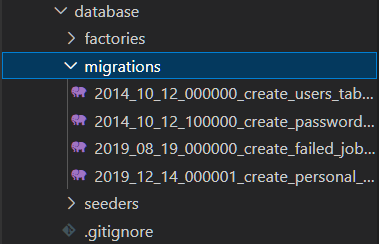
failed\_jobs

migrations

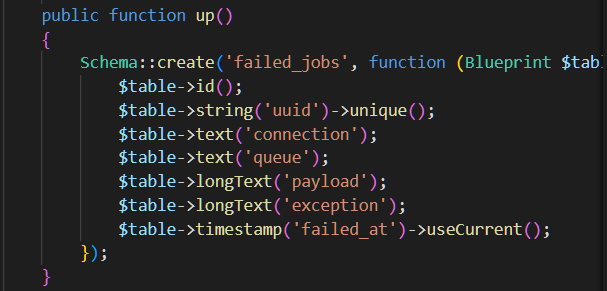
password\_resets

personal\_access\_tokens

users



遷移到資料庫的資料，為Database中的migrations的內部檔案進行定義



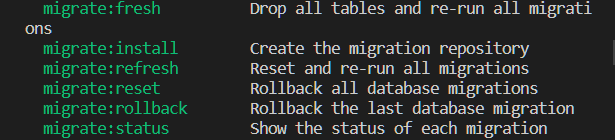
<https://laravel.com/docs/5.2/migrations>

因此，可以不用至Xampp的phpmyAdmin或MySQL WorkBench進行資料庫的操作

只需要至Database/migrations/自定義的php檔案，設置Schema表格需要新增的表格名稱和內容，就可以完成操作

php artisan

顯示可以使用的指令



* Bonus







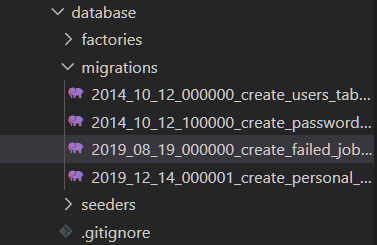


## 透過指令建立遷移檔案

可以使用 make:migration Artisan指令建立遷移：

php artisan make:migration create\_users\_table

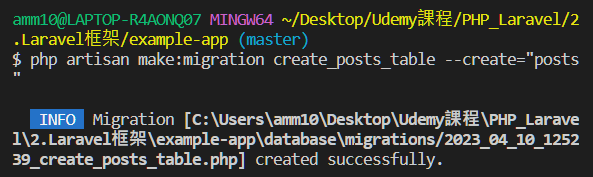
新的遷移檔將會放置在 database/migrations目錄中。每個遷移檔名稱都包含了一個時間戳記，讓 Laravel能夠確認遷移的順序。



--table 和 --create 選項可用來指定資料表的名稱，或是該遷移會建立新的資料表。這些選項只需預先在產生遷移建置檔案時填入指定的資料表：

php artisan make:migration add\_votes\_to\_users\_table --table=users

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



建立一個create\_posts\_table的遷移檔案，並且新增的表為posts

## Rollback & Migrate

    public function up()

    {

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

            $table->id();

            $table->string('title');

            $table->text('body');

            $table->timestamps();

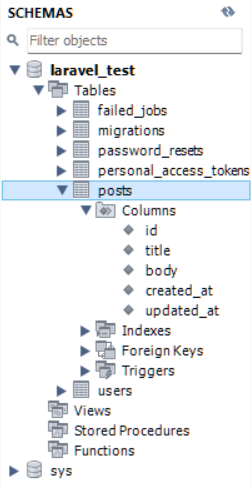
        });

    }

在新建立的遷移檔案中，修改需要新增的表格內容

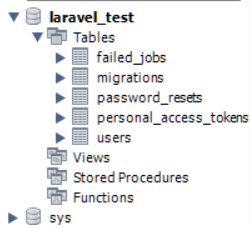
修改完畢後，執行以下指令

php artisan migrate => 遷移資料到資料庫中



自行設置的title和body都出現在資料庫中，代表遷移資料成功

php artisan migrate:rollback => 返回上一步動作(將先前遷移的資料給rollback回朔)



重新整理後，表格消失

rollback執行的是遷移檔案中的down()

    public function down()

    {

        Schema::dropIfExists('posts');

    }

會執行SQL指令，將該表格丟棄

    public function up()

    {

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

            $table->id();

            $table->string('title');

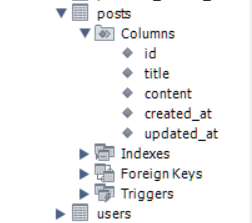
            $table->text('content');

            $table->timestamps();

        });

    }

將body修改成content，再migrate一次



posts表格中的body也成功修改成content

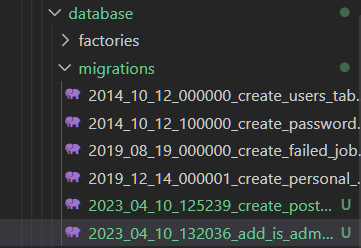
## 新增欄位至已存在的表格

執行以下指令

php artisan make:migration add\_is\_admin\_column\_to\_posts\_table --table=”posts”

add\_is\_admin\_column\_to\_posts\_table => 將要執行的動作用名稱設為遷移檔案的標題，方便確認

--table => 要更改的表格為 “posts” 表格



設置完後會出現新的遷移檔案

    public function up()

    {

        Schema::table('posts', function (Blueprint $table) {

            //

            $table->boolean('is\_admin')->default(0);

        });

    }

在up()內部新增需要增加的欄位名稱和內容

    public function down()

    {

        Schema::table('posts', function (Blueprint $table) {

            //

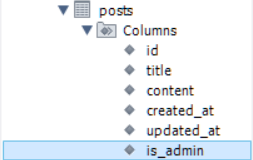
            $table->dropColumn('is\_admin');

        });

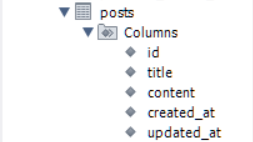
    }

在down()中設置當執行rollback時，會執行的動作

執行php artisan migrate 指令

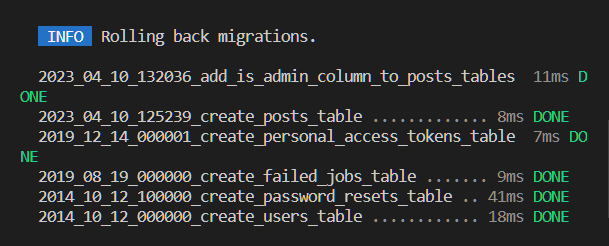


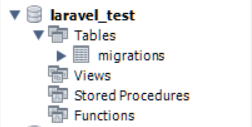
執行php artisan migrate:rollback 指令



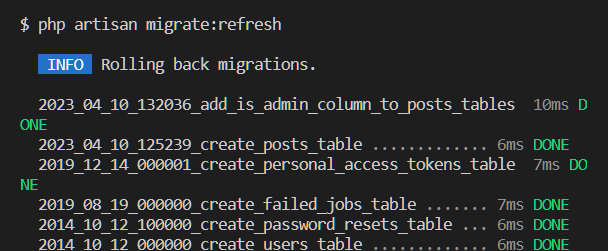
## refresh & reset

php artisan migrate:reset => 將先前所建立的資料給rollback

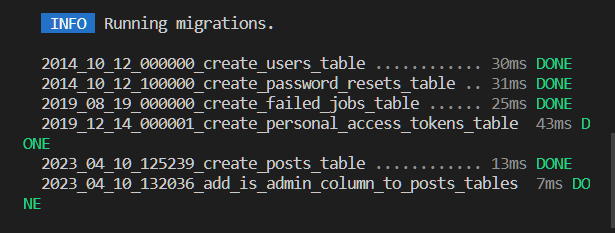




php artisan migrate:refresh => 將所有表格給rollback再migrate



先drop table



再migrate table

