Piccolo manuale Laravel (6.x LTS)

#### Questo manuale vi aiuterà a rinfrescarvi la memoria nei momenti di buio più totale.

## Prerequisiti

### IMPORTANTE!!! **Non** Utilizzate WAMP, XAMP, MAMP, QUACK o qualsiasi altro, munitevi di una Linux Box e assicuratevi di aver installato i seguenti pacchetti/software:

* Git
* Zip
* PHP > 7.2.5
* Composer 2
* npm

Git e Zip sono 2 comandi che generalmente vengono installati con qualsiasi ambiente, per maggiori dettagli sui requisiti visitate il link: <https://laravel.com/docs/6.x> poi fate click sulla voce di menù “Documentation” ed infine su # Installation > Server Requirements (che trovate poco più sopra del centro della pagina).

### Installare Composer

Composer è un gestore di pacchetti per PHP e ci aiuta a gestire le librerie e per installarlo, la Documentazione di Laravel fornisce le indicazioni per tutti i S.O.

Per installare composer su ubuntu, usare il comando:

**$ sudo apt-get install composer**

Seguire la guida: <https://laravel.com/docs/6.x/installation> (e poi proseguite su Installation), potete trovare la sua documentazione completa per cli su: <https://getcomposer.org/doc/03-cli.md>

Assicurarsi che la versione di composer sia la 2.x verificando con il comando

**$ composer --version**

ATTENZIONE!!! Ultimamente, al momento dell’aggiornamento di questo documento (11/05/2021) si rende necessario installare composer 2.0.

Se avete Ubuntu potete seguire questa guida: <https://techstorm.io/how-to-update-or-upgrade-composer-version-in-ubuntu/>

per disinstallare il composer installato di default ed installare quello aggiornato.

I comandi principali sono:

$ **composer require <nomepackage> [vincolo][versione]**

Aggiunge nuove dipendenze aggiungendo librerie al file composer.json

Es: composer require laravel/laravel ^6.0

x.y.z -> Esattamente la versione x.y.z

>=x.y.z -> Versione maggiore o uguale di x.y.z

^x.y.z -> Versione vicina a x.y.z (limite alla major version)

~x.y.z -> Versione vicina a x.y.z (limite alla release)

**$ composer update**

Aggiorna tutte le librerie all'ultima versione disponibile compatibile con quanto indicato in composer.json e al termine crea/aggiorna composer.lock (che contiene le versioni precise di quanto installato).

**$ composer install**

Installa le versioni specificate in composer.lock (che quindi deve già essere creato con un primo composer update).

**$ composer clear-cache**

Pulisce la cache di composer.

## 

## Getting Started!

Se durante la creazione di un nuovo progetto ottenete un errore, controllate nella sezione 5 (Troubleshooting) se è elencato il vostro caso

### Creazione di un progetto Laravel (6.x)

* 1. Creare un nuovo progetto usando uno dei seguenti comandi a seconda se si vuole o meno voler specificare una versione precisa di Laravel:

**composer create-project laravel/laravel --prefer-dist <nome\_progetto>**

**composer create-project laravel/laravel=6 --prefer-dist <nome\_progetto>**

* 1. Configurare il progetto:

**cd <nome\_progetto>**

**cp .env.example .env**

**php artisan key:generate**

**php artisan config:cache**

* 1. (Opzionale) Se il vostro progetto prevede l’utilizzo di un Database creare un nuovo database con MySQL:

CREATE DATABASE <dbname>;

GRANT ALL PRIVILEGES ON <dbname>.\* TO 'username'@'localhost' IDENTIFIED BY 'password';

Configurare il file .env del nuovo progetto con il driver e i parametri di connessione al database

* 1. (Opzionale MA DA FARE COME PRIMA COSA SE RICHIESTO) Se il vostro progetto prevede diversi utenti eseguite i seguenti passaggi la cui spiegazione è disponibile su: <https://github.com/laravel/ui>

**composer require laravel/ui "^1.0" --dev**

**php artisan ui bootstrap --auth**

**npm install && npm run dev**

Maggiori dettagli li potete trovare a questo link: <https://laravel.com/docs/6.x/authentication#authentication-quickstart>

Se avete bisogno di una gestione degli utenti più evoluta (per esempio con i ruoli utente) allora vi consiglio di seguire le indicazioni presenti in [questa discussione su Stack Over Flow](http://stackoverflow.com/questions/23919311/authenticating-a-user-role-in-laravel-and-protecting-a-route)

1. Eseguire il server locale con il seguente comando:

**php artisan serve**

A questo punto visitando il link [**http://127.0.0.1:8000**](http://127.0.0.1:8000), troverete la pagina predefinita “welcome” del vostro progetto

## Convenzioni sui nomi

**Tabelle**: Nome in **minuscolo**, al **plurale** con forma “snake” ( \_ per separare le parole) es. users, tags, ...

**Colonne**: Nome in **minuscolo** al **singolare** con forma “snake” ( \_ per separare le parole) es. firsrt\_name, last\_name, birth\_date...

**Modelli**: Al **singolare** con forma “cammellizzata” (prima lettera in maiuscolo per separare le parole) es. User, Tag, ...

**Controllers**: Al **singolare** con forma “cammellizzata” (prima lettera in maiuscolo per separare le parole) es. UserController, TagController, HelloController ...

## Comandi base

Se durante l’esecuzione dei comandi ottenere un errore, controllate nella sezione 4 (Troubleshooting) se è elencato il vostro caso

### Generatori

// Controller

**php artisan make:controller <nome\_controller\_singolare>Controller**

// Controller completo di tutte le azioni

**php artisan make:controller <nome\_controller\_singolare>Controller --resource**

// Model

**php artisan make:model <nome\_model\_singolare>**

// Model (con migrazione)

**php artisan make:model <nome\_model\_singolare>** **-m**

// Model (con migrazione e controller)

**php artisan make:model <nome\_model\_singolare>** **-m -c --resource**

### Migrazioni

// Creare una migrazione

**php artisan make:migration create\_<nome\_tabella>\_table**

**php artisan make:migration <modifica>\_to\_<nome\_tabella>\_table**

// Eseguire le migrazioni

**php artisan migrate**

// Eseguire il rollback dell’ultima migrazione

**php artisan migrate:rollback [--step=n]**

// Eseguire il reset del database, cancella tutte le tabelle e riesegue le migrazioni

**php artisan migrate:fresh**

### Seeder

// Creare un seeder

**php artisan make:seeder <nome\_model\_plurale>TableSeeder**

// Eseguire tutti i seed

**php artisan db:seed**

// Eseguire il seed solo di una specifica classe

**php artisan db:seed --class=<nome\_model\_plurale>TableSeeder**

Potrebbe capitare, durante l’esecuzione dei Seed di ricevere l’errore:

**Target class [UsersTableSeeder] does not exist.**

Questo succede perchè al progetto sono stati aggiunti dei file (i seeder appunto) che non riescono ad essere identificati automaticamente.

Per risolvere il problema bisogna eseguire il comando:

**composer dump-autoload**

Che andrà ad indicizzare i file aggiunti.

## Troubleshooting

**Errori durante la creazione del progetto su Ubuntu**

Installate i seguenti pacchetti con i comandi:

sudo apt-get install php-mbstring

sudo apt-get install php-xml

**Errori durante la creazione del progetto**

Può capitare che durante la creazione del progetto il comando restituisca un errore simile a:

“Script php artisan clear-compiled handling the pre-update-cmd event returned with error code 255”

Per risolvere questo errore cancellate la dir del progetto ed eseguite il seguente comando per cancellare la cache di composer:

**composer clear-cache**

A questo punto ripartite dal punto 1a o 1b.

**Errori nella lettura del config**

Se ricevete degli errori che su delle configurazioni che non rispettano quello che avete inserito nel file .env allora dovete eseguire la pulizia della cache del file di config:

php artisan config:clear

**Errore relativo a sass (con molto testo in rosso) durante l’esecuzione di npm**

npm uninstall --save-dev sass-loader

npm install --save-dev sass-loader@7.1.0

E poi di nuovo

npm install && npm run dev

**Il codice JS o CSS non esegue quello che volete o non trova funzioni che avete appena scritto**

Eseguire la pulizia della cache delle view:

php artisan view:clear

**Errori che sembrano venire da codice “vecchio” che avete rimosso**

Eseguire la pulizia della cache:

php artisan cache:clear

**Errore SQL 42000 durante le migrazioni**

Durante l’esecuzione delle migrazioni (generalmente alla prima) appare un errore riguardante una *“Key too long; max length is ...”*

Per risolverlo editate il file AppServiceProvider.php e aggiungete in testa:

use Illuminate\Support\Facades\Schema;

e nella funzione boot() inserite la seguente riga di codice:

Schema::defaultStringLength(191);

A questo punto cancellate le tabelle e rieseguite le migrazioni

## Materiale

Questo link contiene una serie di video-lezioni in inglese (ma molto facili da seguire) che possono velocizzare l’apprendimento:

<https://laracasts.com/series/laravel-6-from-scratch>

È fondamentale arrivare al video 41 (tutta la sezione 10)

## 

## Piccolo manuale di Vue-JS

Video per le funzionalità base:

<https://www.youtube.com/watch?v=U2iZj_txFSc>

Video per l’utilizzo con Laravel 6 (con i component)

<https://www.youtube.com/watch?v=DJ6PD_jBtU0>

## Deploy di un progetto su Heroku

Dopo essersi registrati su <https://www.heroku.com/> installare la Heroku toolbelt seguendo il tutorial a questo link: <https://devcenter.heroku.com/articles/heroku-command-line>

### Deploy dell’applicazione

Inizializzare un repository locale

**git init**

**git add .**

**git commit -m ...**

Eseguire login su Heroku

**heroku login**

Creare una nuova App Heroku

**heroku create <nome\_app>**

Aggiungere il pack per PHP (su un’ unica riga)

**heroku buildpacks:set https://github.com/heroku/heroku-buildpack-php**

Aggiungere il Procfile

**echo "web: vendor/bin/heroku-php-apache2 public/" >> Procfile**

Eseguire il commit delle modifiche

**git add .**

**git commit -m “Setup per deploy”**

Eseguire il push del repository su Heroku

**git push heroku master**

**heroku config:set APP\_KEY=$(php artisan --no-ansi key:generate --show)**

Aprire il browser direttamente sulla vostra applicazione

**heroku open**

### Preparazione del Database

Creare un nuovo database su Heroku utilizzando l’opzione free (hobby)

**heroku addons:create heroku-postgresql:hobby-dev**

Ottenere le credenziali per la connessione al database

ATTENZIONE: il nome del database lo ottenete dall’output del comando addons:create oppure usando il comando pg:info. Le informazioni verranno visualizzate come stringa unica

**heroku config | grep DATABASE\_URL**

Inserite la configurazione del database di produzione

Nel file config/database.php del progetto configurare il driver pgsql e i parametri per il database di produzione

Eseguire il commit delle modifiche

**git add -u**

**git commit -m “Configurazione db produzione”**

Eseguire il push del repository su Heroku

**git push heroku master**

Aprire una shell sull’account Heroku

**heroku run bash**

Eseguire le migrazioni ed i seed

**php artisan migrate**

**php artisan db:seed**

### Download del backup di produzione

È utile avere una copia di backup del database di produzione

Eseguire uno snapshot del database

**heroku pg:backups capture**

Scaricare la copia dell’ultimo backup

**curl -o latest.dump `heroku pg:backups public-url`**

È spesso utile poter lavorare in locale con un copia dei dati in produzione per poter avere dati “reali” mentre si sviluppa e si fanno i test.

Dopo aver scaricato in locale la copia del database, si può eseguire il restore del backup sul database locale

Restore database

**pg\_restore --verbose --clean --no-acl --no-owner -h localhost -U <user\_name> -d <database\_name> latest.dump**

### Altri comandi utili

Per aprire il browser direttamente sulla vostra applicazione

**heroku open**

Controllare i log del server su Heroku

**heroku logs**

Ottenere le informazioni sul database

**heroku pg:info**

Elenco degli addon installati

**heroku addons**

### Plugin utili da installare

Editor Vim da usare sul vostro account

**heroku plugins:install** [**https://github.com/naaman/heroku-vim**](https://github.com/naaman/heroku-vim)

### Backup settimanale

Backup settimanale (da verificare!)

**heroku addons:add pgbackups:auto-week**