boolean

CORSO FULL STACK WEB DEVELOPER

LARAVEL

Migrations

Con Laravel possiamo creare e modificare le tabelle del database

Creare le tabelle con Laravel

L'unica cosa che dovremo fare con tool esterni, come ad esempio phpMyAdmin, sarà la creazione del database.

Dopo aver inserito i dettagli di connessione al database nel fine .env, **potremo scrivere dentro Laravel il codice php** che andrà a generare le tabelle!

Usiamo il comando da console

```
1 php artisan make:migration nome_della_migration
```

Migration

Create

Se diamo un nome alla nostra migration rispettando un certo tipo di schema, Laravel scriverà per noi buona parte del codice.

Con questo comando, ad esempio, viene creata una migration che prende in automatico il nome della tabella (users), inserisce l'id e altri dati.

```
1 php artisan make:migration create_users_table
```

Migration

Update

Per **aggiornare un tabella** possiamo usare invece questo comando.

L'opzione --table indica a Laravel qual è la tabella da modificare, così da avere anche in questo caso parte del codice già compilato.

```
1 php artisan make:migration update_users_table --table=users
```

Migration

Una migration è composta da due parti (che in realtà sono due metodi) indispensabili.

1 **Up**

Nel metodo up() si inseriscono i comandi da eseguire quando viene eseguita una migration.

Solitamente sono comandi che creano o modificano una tabella, o in generale fanno un *upgrade* di qualsiasi tipo.

2 **Down**

Il metodo down() è l'opposto di up(), viene eseguito quando si vuole annullare una migration e il suo scopo è eliminare ciò che up() ha fatto.

Se up() ha creato una tabella, down() la cancella, se up() ha modificato una colonna, down() la farà tornare allo stadio precedente.

Down

Ma serve davvero il metodo down?

Il metodo down() è indispensabile per poter tornare indietro qualora le modifiche effettuate sul database con una up() non andassero bene.

Up Base

Schema è la **classe Laravel** che si occupa di creare la tabella.

In questo esempio, con il metodo **create()** viene creata una tabella **flights**, con le seguenti colonne:

id incrementale,
name di tipo stringa
airline di tipo stringa

Il metodo **timestamps()** invece creerà per noi due colonne:
created_at e **updated_at** di tipo DateTime.

```php
public function up() {
    Schema::create('flights', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('name');
        $table->string('airline');
        $table->timestamps();
    });
}
```

Quando utilizziamo il comando artisan per creare una tabella, Laravel inserirà per noi l'id e i timestamps.

L'id sarà già considerata la chiave primaria, senza bisogno di indicare altro.

Down Base

Il metodo **down()** utilizza sempre la classe **Schema.**

In questo esempio, tramite il metodo **drop()** viene cancellata la tabella flights, creata nel metodo **up()**.

```php
public function down(){
    Schema::drop('flights');
}
```

Eseguire le Migration

Per fare un upgrade delle tabelle ed eseguire tutte le migration **non ancora eseguite**

```
1 php artisan migrate
```

Se vogliamo invece fare un **downgrade** delle nostre migration e tornare allo stato precedente rispetto l'ultima esecuzione delle migration

```
1 php artisan migrate:rollback
```

Le migration eseguite o annullate contemporaneamente potranno essere anche più di una!

Questo perché Laravel tiene traccia di quali migration sono state eseguite e quando.
Tutte quelle non ancora eseguite verranno eseguite, mentre il rollback funzionerà solo sull'ultima serie di migration eseguite.

Cancellare tutte le Migration

Attenzione! Verranno cancellate tutte le nostre migration e quindi tutti i dati!!!

```
1 php artisan migrate:reset
```

Come agisce Laravel

Abbiamo visto che Laravel ricorda quali migration ha eseguito e quali mancano ancora per gestire sia il migrate che il rollback, ma come fa?

All'inizio si crea nel database una tabella che prende il nome di migrations in cui si salva il nome della migration e in che serie è stata eseguita.

In questo modo potrà agire anche sul rollback. Prenderà le migrations con batch più alto e farà down di esse.

+ Opciones

migration	batch
2014_08_21_050304_sectores	1
2014_08_21_050559_clientes	1
2014_08_21_051357_licencias	1
2014_08_21_051638_usuarios	1
2014_08_21_125458_basesDeDatos	1
2014_08_21_162042_carpetas	1
2014_08_21_165225_permisos_carpetas	1
2014_08_27_021433_subscripciones	1
2014_08_27_032816_modulos	1
2014_08_30_233714_licencias_modulos	1
2014_08_31_003753_modulos_usuario_view	1
2014_08_31_014413_permisos_arquitectura	1
2014_08_31_231138_usuario_database_view	1
2014_09_02_025032_usuarios_configuraciones	1
2014_09_03_031439_usuarios_cliente_view	1
2014_09_07_172624_permisosCompromisos	1
2014_09_07_173411_permisos_reuniones	1
2014_09_07_185357_permisos_proyectos	1
2014_09_07_205010_permisos_bsc	1
2014_09_11_050011_permisos_generales	1
2014_09_25_170912_etiqueta	1
2014_09_25_201920_clase_compromiso	1

LIVE CODING

Creiamo una tabella con una migration

ESERCITAZIONE