MIGRATIONS AND DATABASE IN LARAVEL

CNA FORMAZIONE
VIA MULINI 35,
CESENA (FC)

18 - NOVEMBRE - 2015

MIGRATIONS

Sono una specie di controllo di versione del database. Sono delle classi PHP che scriviamo per creare, modificare e/o cancellare tabelle per gestire i dati delle applicazioni.
Sono eseguite in serie al nostro comando, e Laravel si occupa di eseguire solo i nuovi aggiornamenti e ci mette anche a disposizione uno strumento per tornare indietro nel caso in cui avessimo commesso un errore nell'aggiornamento.

MIGRATIONS

Sono una specie di controllo di versione del database. Sono delle classi PHP che scriviamo per creare, modificare e/o cancellare tabelle per gestire i dati delle applicazioni.
Sono eseguite in serie al nostro comando, e Laravel si occupa di eseguire solo i nuovi aggiornamenti e ci mette anche a disposizione uno strumento per tornare indietro nel caso in cui avessimo commesso un errore nell'aggiornamento.

TABELLE USERS - MIGRATIONS

Alla creazione dell'app sono già presenti due migrations per creare le tabelle relative agli utenti.

Assicurarsi di aver correttamente configurato la connessione al database prima di eseguire qualunque operazione relativa a migrations and database.

USO MIGRATIONS

Per verificare lo stato attuale delle migrazioni con la shell:
php artisan migrate:status

Per eseguire le ultime migrazioni non ancora eseguite:
php artisan migrate

Per tornare indietro prima dell'ultima migration:
php artisan migrate:rollback

NUOVA MIGRATION: CREARE UNA TABELLA

Dalla shell (è un unico comando, va in una sola riga):
php artisan make:migration create_articles_table
--create="articles"

Verrà creata la nuova migration in database/migrations/

Altre possibili modalità per una migrazione:
php artisan help make:migration

NUOVA MIGRATION: CREARE UNA TABELLA

La migration appena creata può essere ampliata così:

```
$table- >increments('id');
$table- >string('title');
$table- >text('body');
$table- >timestamps();
$table- >timestamp('published_at');
```

MODIFICARE UNA TABELLA

Due possibilità:

1) Rollback della migration, cambiare il codice della migration su cui fare la modifica, rieseguirla. Si può fare così solo se non si è già distribuita ed eseguita la migration incriminata in altri ambienti!

2) Fare una nuova migration che modifica la precedente! Come?

php artisan make:migration add_new_col --table="articles"

MODIFICARE UNA TABELLA

Per la nuova migration va inserito il codice di cambiamento.
Per esempio, per aggiungere una nuova colonna, nella funzione up() aggiungeremo il seguente codice:
$table- >text('excerpt')- >nullable();

Sempre nella stessa migration, invece, per la funzione down():
$table- >dropColumn('excerpt');
Ricordarsi però di eseguire: composer require doctine/dbal

ELOQUENT MODEL IN LARAVEL

Un Model è un oggetto PHP che serve a mappare una riga di una tabella. Per esempio, se abbiamo la tabella Articles possiamo creare una classe Article che rappresenta una riga di quella tabella.

Per la creazione, da riga di comando eseguire:
php artisan make:model Article
Creerà nella cartella app un file di nome Article.php

TINKER

Giochiamo un po' con una sessione interattiva di Laravel chiamata Tinker, per imparare ad usare il Model Article appena creato.

```
$article = new App\Article;
$article- >title = 'Titolo';
$article- >body = 'Descrizione';
$article- >published_at = Carbon\Carbon::now();
```

TINKER

Vediamo il contenuto dell'oggetto:

$article;

$article- >toArray();

Salviamo ora?

$article- >save();

TINKER

Recuperiamo ora l'oggetto salvato leggendo dal database, in modo da verificare se abbiamo effettivamente salvato i dati:

```
App\Article::all()- >toArray();
$art = App\Article::find(1); // 1 è l'id se lo conosciamo
$art = App\Article::where('body', 'Descrizione')- >get(); // tutti
$art = App\Article::where('body', 'Descrizione')- >first(); // il 1°
```

TINKER

Alternativa di creazione con un unico comando:
$article = App\Article::create(['title' => 'Titolo', 'body' => 'Descrizione', 'published_at' => Carbon\Carbon::now()]);
Ricordiamoci però di aggiungere alla classe Article:
protected $fillable = ['title', 'body', 'published_at'];

Per aggiornare o modifichiamo il campo e poi save() oppure:
$article- >update(['title' => 'Titolo AGGIORNATO']);

GRAZIE

per l'attenzione!