Les migrations de base de données Laravel

1. Objectifs

- Etre capable de migrer une base de données avec Laravel.

2. Présentation

- Une migration permet de créer et de mettre à jour un schéma de base de données. Autrement dit, vous pouvez créer des tables, des colonnes dans ces tables, en supprimer, créer des index... Tout ce qui concerne la maintenance de vos tables peut être pris en charge par cet outil. Vous avez ainsi un suivi de vos modifications.
- La migration de base de données est l'une des meilleures fonctionnalités fournies par **Laravel**.
- Dans le passé, vous devez créer une base de données, une table et des colonnes soit par codage **SQL** ou en utilisant un outil comme **PHPMyAdmin**. La tâche semble parfois intimidante.
- De plus, nous avons un avantage supplémentaire. En tant que développeur, travaillant en équipe, vous pouvez obtenir votre travail de base de données bien synchronisé avec vos collègues. En ce sens, c'est devenu comme un contrôle de version. Vous pouvez facilement déployer votre table de base de données et mettre à jour avec de nouvelles fonctionnalités via des codes php dans votre application.
- Tout ce que vous avez à faire est d'écrire quelques lignes de code PHP et Laravel s'occupera du prochain pas.

3. Que sont les migrations

> *Vous ne devez jamais travailler manuellement dans votre outil de gestion de bases de données pour ajouter une table ni pour en modifier la structure.* ***TOUT DOIT PASSER PAR LES FICHIERS DE MIGRATION****.*

- La première étape du développement d'une application Web consiste à concevoir une base de données. Laravel offre un excellent moyen de concevoir des schémas et des tables de base de données et de les migrer facilement sur différents systèmes appelés "**migrations**".

apcpedagogie.com

- Les migrations sont utilisées pour créer, modifier et partager le schéma de base de données de l'application. Ils sont utilisés avec le générateur de schéma de Laravel pour créer facilement un schéma de base de données.
- Les migrations sont une sorte de carnet de bords de toutes les modifications qui peuvent être effectuées dans l'organisation de votre base de données (création d'une table, ajout, suppression ou renommage de colonne dans une table etc...).
- Les migrations sont des fichiers php que vous pouvez retrouver directement dans vos dossiers, elles se situent dans le dossier : **/database/migrations/**
- En vous rendant dans ce dossier vous découvrez que 2 migrations existent déjà. Une pour la création de la table **users** et une pour la table **password_resets**.

- Le contenu du fichier **`create_users_table.php`** est le suivant :

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
```

apcpedagogie.com

```
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

4. Concernant les méthodes de la Class `CreateUsersTable` il n'y en a deux :
    - La méthode **`up()`** qui permet d'ajouter de nouvelles tables, colonnes, index etc...
    - La méthode **`down()`** qui permet de faire un "retour en arrière" de toutes les opérations effectuer par la méthode **up()**.

5. Les commandes (artisan) utilisées dans la migration

- Laravel est livré avec un ensemble de commandes liées à la migration qui peuvent être exécutées à partir d'une interface de ligne de commande artisan.

- **migrate: install**
    - Cette commande permet de créer la table de migration dans la base de données de l'application, cette table permet de suivre les migrations déjà exécutées.

- **migrate**
    - Cette commande est utilisée pour exécuter toutes les migrations en attente pour votre application.

- **migrate: make**
    - Cette commande est utilisée pour créer une nouvelle migration, la migration nouvellement créée est enregistrée dans le répertoire base de données / migrations. Un horodatage sera ajouté au début du nom du fichier de migration, ce qui permet à Laravel de déterminer l'ordre des migrations.

- **migrate: rollback**
    - Cette commande est utilisée pour annuler la dernière migration appliquée.

- **migrate: reset**

apcpedagogie.com

- Cette commande permet d'annuler toutes les migrations appliquées.

- **migrate: refresh**

  - Cette commande est utilisée pour annuler toutes vos migrations de base de données, puis recrée votre schéma de base de données.

6. Créer une migration avec artisan

- **Artisan**

  - Nous avons déjà utilisé Artisan qui permet de faire beaucoup de choses, vous avez un aperçu des commandes en entrant :**`php artisan`**

  - l'image ci-dessus présente les commandes suivantes:
    - **`fresh`**: supprime toutes les tables et relance la migration (commande apparue avec la version 5.5)
    - **`install`** : crée et informe la table de référence des migrations
    - **`refresh`** : réinitialise et relance les migrations
    - **`rollback`** : annule la dernière migration
    - **`status`** : donne des informations sur les migrations
- Pour contrôler votre base de données dans Laravel, utilisez les migrations. Créer une migration avec artisan:
- **`php artisan make:migration create_utilisateurs_table --create=utilisateurs`**
- En exécutant, cette commande le fichier **app/database/migration/_create_utilisateurs_table.php** a été créé.

- Son contenu est le suivant :

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUtilisateursTable extends Migration
{
    /**
     * Run the migrations.
     *
```

apcpedagogie.com

```
     * @return void
     */
    public function up()
    {
        Schema::create('utilisateurs', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('utilisateurs');
    }
}
```

7. Cette dernière commande génère un template de source PHP qui permettra de créer une table "**utilisateurs**" dans la base de données, si nous migrons ce script en base de données via la commande suivante :`php artisan migrate`
8. Cette commande lance les migrations, des fichiers crées dans le dossier **migration**

```
PS C:\wamp64\www\laravel\monProjet> php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (686.62ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (724.83ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (631.62ms)
Migrating: 2020_12_23_075316_create_utilisateurs_table
Migrated:  2020_12_23_075316_create_utilisateurs_table (243.56ms)
```

9. Pour annuler ou rafraichir une migration on utilise la commande rollback
10. Si vous rencontrez l'erreur suivant:

11.
```
Migration table created successfully.

[Illuminate\Database\QueryException]
```

apcpedagogie.com

```
SQLSTATE[42000]: Syntax error or access violation: 1071 La cl&#9500;® est trop longue.
Longueur maximale: 1000 (SQL: alter
table `users` add unique `users_email_unique`(`email`))

[PDOException]
SQLSTATE[42000]: Syntax error or access violation: 1071 La cl&#9500;® est trop longue.
Longueur maximale: 1000
```

- Avec MySql5.6.XXX, une solution plus clean consiste à modifier le fichier **database.php** dans le dossier config :
  - `'engine' => null,`
  - Devient :`'engine' => 'InnoDB ROW_FORMAT=DYNAMIC',`
  - Ou tout autre type de engine paramétré lors de l'install de MySql...

12. Créer une migration d'un seul fichier

- Il existe une commande d'artisan pour créer une squelette de migration pour un seul fichier:

- Puis exécutez la commande suivante:`php artisan migrate --path=database/migrations/2020_12_23_094127_utilisateurs.php`

13. Annuler ou rafraichir une migration

- Pour annuler une migration on utilise la commande `rollback`: `PHP artisan migrate:rollback`
- Exécuter **migrate:rollback-** c'est la bonne façon d'annuler la dernière migration.
  Si **migrate:rollback** ne fonctionne pas, faites-le manuellement:

1. Supprimez manuellement le fichier de migration sous app/database/migrations/my_migration_file_name.php
2. Réinitialisez les fichiers de chargement automatique du composeur: composer dump-autoload
3. Modifier votre base de données: supprimez la dernière entrée de la table des migrations

14. Application

- Créer, en utilisant une seule commande, un Model nommé **Client** ainsi qu'une Migration permettant de créer la table correspondant au Model
- Ajouter, dans la Migration, le code nécessaire permettant de créer la table dans la BD sachant que la table est représentée de la façon suivante: **clients (id, nom, prenom,**

apcpedagogie.com

dateNaissance, adresse, tel)

- Lancer la commande permettant d’exécuter la Migration.

Cours Laravel

Source : https://apcpedagogie.com/les-migrations-de-base-de-donnees-laravel · téléchargé le 06/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com