TP N 1
Base de données en Laravel

ISTA TINGHIR
Realisé par: Groupe 1
le: 6 maj 2023
Module: M205
Développer en Back-End

Introduction

Presque toutes les applications web modernes interagissent avec une base de données. Laravel rend l'interaction avec les bases de données extrêmement simple sur une variété de bases de données prises en charge en utilisant SQL brut, un constructeur de requêtes fluide et l'ORM Eloquent. Actuellement, Laravel prend en charge en interne cinq bases de données : MariaDB, MySQL ,PostgreSQL, SQLite ,SQL Server

Configuration de la base de données

La configuration des services de base de données de Laravel se trouve dans le fichier de configuration config/database.php config/database.php de votre application. Dans ce fichier, vous pouvez définir toutes vos connexions de base de données, ainsi que spécifier quelle connexion doit être utilisée par défaut. La plupart des options de configuration dans ce fichier sont basées sur les valeurs des variables d'environnement de votre application. Des exemples pour la plupart des systèmes de base de données pris en charge par Laravel sont fournis dans ce fichier.
Fichier : config/database.php

```php
<?php

use Illuminate\Support\Str;

return [

    'default' => env('DB_CONNECTION', 'mysql'),


    'connections' => [

        'sqlite' => [
            'driver' => 'sqlite',
            'url' => env('DATABASE_URL'),
            'database' => env('DB_DATABASE',
               database_path('database.sqlite')),
            'prefix' => '',
            'foreign_key_constraints' => env('DB_FOREIGN_KEYS', true),
        ],

        'mysql' => [
            'driver' => 'mysql',
```

```
            'url' => env('DATABASE_URL'),
            'host' => env('DB_HOST', '127.0.0.1'),
            'port' => env('DB_PORT', '3306'),
            'database' => env('DB_DATABASE', 'forge'),
            'username' => env('DB_USERNAME', 'forge'),
            'password' => env('DB_PASSWORD', ''),
            'unix_socket' => env('DB_SOCKET', ''),
            'charset' => 'utf8mb4',
            'collation' => 'utf8mb4_unicode_ci',
            'prefix' => '',
            'prefix_indexes' => true,
            'strict' => true,
            'engine' => null,
            'options' => extension_loaded('pdo_mysql') ? array_filter([
                PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
            ]) : [],
        ]
```

2- Dans le fichier .env, définissez les informations de connexion à votre base de données. Par exemple :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mydatabase
DB_USERNAME=myusername
DB_PASSWORD=mypassword
```

Connexion a la base de données

Pour établir une connexion à la base de données, vous pouvez utiliser la façade DB. Par exemple, pour sélectionner toutes les lignes de la table users :

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class userController extends Controller
{
    //afficher la liste de tous les utilisateurs
    public function index(){
        $users = DB::table('users')->get();
        //return view('users.index',compact('users'));
        dd($users);
        //Vous pouvez  galement  utiliser Eloquent, le ORM int gr
           de Laravel
        $users = User::all();
        dd($users);
    }

}
```

Exécution de requêtes SQL (requêtes CRUD)

la requête SELECT

Le premier argument passé à la méthode select est la requête SQL, tandis que le deuxième argument est tout paramètre à lier à la requête. En général, il s'agit des valeurs des contraintes de la clause WHERE. La liaison de paramètres assure une protection contre les injections SQL.

```
$users = DB::select('select * from users where id = ?', [1]);
```

On peut aussi utiliser la syntaxe suivante:

```
$results = DB::select('select * from users where id = :id', ['id' =>
    1]);
```

la requête INSERT

Pour exécuter une instruction d'insertion, vous pouvez utiliser la méthode insert sur la façade DB. Comme pour la méthode select, cette méthode accepte la requête SQL en tant que premier argument et les liaisons en tant que deuxième argument :

```
DB::insert('insert into users (id, name) values (?, ?)', [6, 'Jhon']);
```

la requête update

La méthode update doit être utilisée pour mettre à jour les enregistrements existants dans la base de données. Le nombre de lignes affectées par l'instruction est retourné par la méthode :

```
$updated = DB::update(
    'update users set name = "Marc" where id = ?',
    [1]);
```

la requête delete

La méthode delete doit être utilisée pour supprimer des enregistrements de la base de données. Comme pour la méthode update, le nombre de lignes affectées sera retourné par la méthode :

```
$deleted = DB::delete('delete from users where id=?',[2]);
```

Gestion des transactions

Que ce que une transaction de base de données?

Une transaction de base de données en Laravel est un groupe d'instructions SQL qui sont exécutées de manière atomique, ce qui signifie qu'elles sont toutes exécutées ou aucune ne l'est. Une transaction permet de garantir l'intégrité des données en s'assurant que toutes les instructions SQL sont exécutées avec succès ou qu'aucune n'est exécutée du tout. Cela peut être particulièrement utile lorsqu'on effectue des opérations qui doivent être traitées ensemble de manière cohérente, par exemple lorsqu'on met à jour plusieurs tables en même temps. En utilisant des transactions, on peut s'assurer que les données de la base de données restent cohérentes et éviter les problèmes de concurrence.

```
use Illuminate\Support\Facades\DB;

DB::transaction(function () {
    DB::update('update users set name = "Devine" where id=?',[3]);
    DB::delete('delete from Archive');
});
```

Utilisation manuelle des transactions L'utilisation manuelle des transactions de base de données en Laravel consiste à entourer les opérations de base de données dans une transaction et à gérer manuellement le commit ou le rollback en fonction du résultat de ces opérations.
Exemle :

```
    DB::beginTransaction();

try {
    $updated = DB :: update (
    update users set name = " Marc " where id = ?      ,
[1]);

    DB::delete('delete from archive where id=?',[1]);

    DB::commit();
} catch (\Exception $e) {
    DB::rollback();
}
```

Connexion à la base de données par l'Interface en ligne de commande (CLI).

Si vous souhaitez vous connecter à l'interface de ligne de commande de votre base de données, vous pouvez utiliser la commande Artisan db":

```
    php artisan db
```

Si nécessaire, vous pouvez spécifier un nom de connexion de base de données pour vous connecter à une connexion de base de données qui n'est pas la connexion par défaut :

```
    php artisan db mysql
```