

Laravel - Database

Topics : [Laravel](#)

Written on [December 25, 2023](#)

In Laravel, working with databases is made easy through the Eloquent ORM (Object-Relational Mapping) and the underlying Query Builder. Laravel supports multiple database systems, including MySQL, PostgreSQL, SQLite, and SQL Server. Here's a brief overview of working with databases in Laravel:

1. Configuration:

Database configuration is defined in the `config/database.php` file. You can set the connection details for different environments (development, testing, production) in the `.env` file.

2. Eloquent ORM:

Eloquent is Laravel's implementation of the active record pattern, allowing you to interact with your database using PHP objects. Each table in your database has a corresponding "model" in Laravel.

Creating a Model:

You can create a model using the Artisan command:

```
php artisan make:model ModelName
```

Example Model:

```
namespace App;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    // Model-specific code
}
```

Example Usage:

```
// Retrieving all users
$users = User::all();
```

```
// Finding a user by primary key
$user = User::find(1);

// Querying with conditions
$users = User::where('name', 'John')->get();

// Creating a new user
$newUser = new User;
$newUser->name = 'Jane';
$newUser->email = 'jane@example.com';
$newUser->save();
```

3. Query Builder:

Laravel also provides a query builder for direct database queries:

```
// Select query
$users = DB::table('users')->get();

// Insert query
DB::table('users')->insert([
    'name' => 'John Doe',
    'email' => 'john@example.com',
]);

// Update query
DB::table('users')->where('id', 1)->update(['name' => 'New Name']);

// Delete query
DB::table('users')->where('id', 1)->delete();
```

4. Migrations:

Migrations allow you to version-control your database schema. You can create and run migrations to modify database tables.

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

Example migration file:

```
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamps();
    });
}
```

Run the migration:

```
php artisan migrate
```

5. Seeding:

Database seeding is the process of populating your database with sample data. Laravel provides a simple way to seed your database using seed classes:

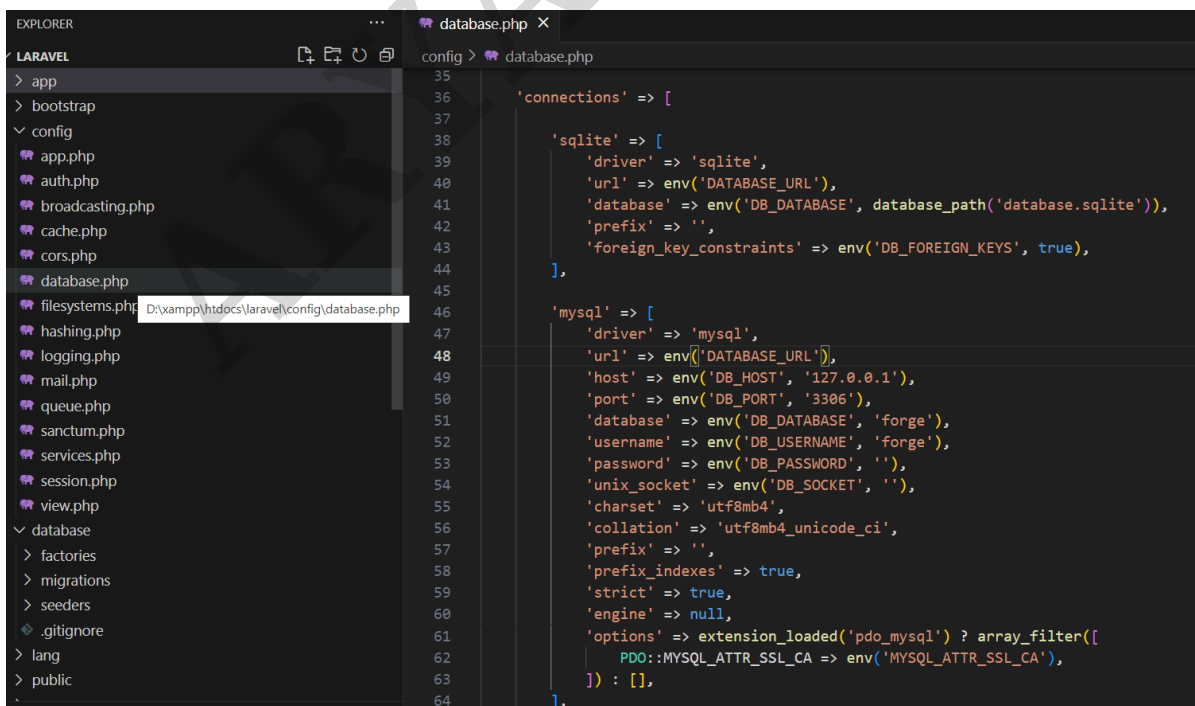
```
php artisan make:seeder UsersTableSeeder
```

Example seeder:

```
public function run()
{
    DB::table('users')->insert([
        'name' => 'John Doe',
        'email' => 'john@example.com',
    ]);
}
```

Run the seeder:

```
php artisan db:seed --class=UsersTableSeeder
```



```
EXPLORER
LARAVEL
> app
> bootstrap
> config
  app.php
  auth.php
  broadcasting.php
  cache.php
  cors.php
  database.php
  filesystems.php
  hashing.php
  logging.php
  mail.php
  queue.php
  sanctum.php
  services.php
  session.php
  view.php
  database
  factories
  migrations
  seeders
  .gitignore
  lang
  public
  database.php X
config > database.php
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
'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'),
        ]) : [],
```