

Pemograman Web

2022/2023 Genap

Febri Damatraseta Fairuz, S.T., M.Kom

01

Website Databases

Migration and Eloquent ORM



**Kampus
Merdeka**
INDONESIA JAYA

Migrations are like version control for your database, allowing your team to define and share the application's database schema definition. If you have ever had to tell a teammate to manually add a column to their local database schema after pulling in your changes from source control, you've faced the problem that database migrations solve.

Laravel Migrations



Configurations

The configuration for Laravel's database services is located in your application's `config/database.php` configuration file. In this file, you may define all of your database connections, as well as specify which connection should be used by default. Most of the configuration options within this file are driven by the values of your application's environment variables. Examples for most of Laravel's supported database systems are provided in this file.

```
23-24-GENAP-PW
├── app
├── bootstrap
├── config
├── database
├── public
├── resources
├── routes
├── storage
├── tests
├── .editorconfig
├── .env
├── .env.encrypted
├── .env.example
├── .gitattributes
└── .gitignore
```

```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=db_bedtime_stories
15 DB_USERNAME=root
16 DB_PASSWORD=
```

By default, Laravel's sample **environment configuration** is ready to use with Laravel Sail, which is a Docker configuration for developing Laravel applications on your local machine.

However, you are free to modify your database configuration as needed for your local database.



Generating Migrations

You may use the `make:migration` Artisan command to generate a database migration. The new migration will be placed in your database/migrations directory. Each migration filename contains a timestamp that allows Laravel to determine the order of the migrations:

Syntax:

```
php artisan make:migration table_name
```

Sample:

```
php artisan make:migration books
```

```
23-24-GENAP-PW
> app
> bootstrap
> config
> database
  > factories
  > migrations
    2023_04_07_082231_create_b... 4, U
  > seeders
> .gitignore
> public
> resources
> routes
> storage
> tests
.editorconfig
.env
.env.encrypted
.env.example
.gitattributes
.gitignore
.htaccess
artisan
composer.json
composer.lock
package.json
phpunit.xml
README.md
vite.config.js

database > migrations > 2023_04_07_082231_create_books_table.php > ...
Click here to ask Blackbox to help you code faster
<?php
1
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('books', function (Blueprint $table) {
15             $table->id();
16             $table->string('title',50);
17             $table->string('author',50);
18             $table->string('synopsis',100);
19             $table->text('story');
20             $table->integer('rate');
21             $table->text('image_cover');
22             $table->tinyInteger('is_active');
23             $table->timestamps();
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('books');
33     }
34 };
```

Type Data:

bigIncrements	ipAddress	timeTz
bigInteger	json	time
binary	jsonb	timestampTz
boolean	longText	timestamp
char	macAddress	timestampsTz
dateTimeTz	mediumIncrements	timestamps
dateTime	mediumInteger	tinyIncrements
date	mediumText	tinyInteger
decimal	morphs	tinyText
double	nullableMorphs	unsignedBigInteger
enum	nullableTimestamps	unsignedInteger
float	nullableUuidMorphs	unsignedMediumInteger
foreignId	nullableUuidMorphs	unsignedSmallInteger
foreignIdFor	rememberToken	unsignedTinyInteger
foreignUuid	set	uuidMorphs
foreignUuid	smallIncrements	uuidMorphs
geography	smallInteger	uuid
geometry	softDeletesTz	uuid
id	softDeletes	year
increments	string	
integer	text	



Running Migrations

To run all of your outstanding migrations, execute the migrate Artisan command:

```
php artisan migrate
```

#Rolling Back Migrations

```
php artisan migrate:rollback
```

```
php artisan migrate:reset
```



db_bedtime_stories

books



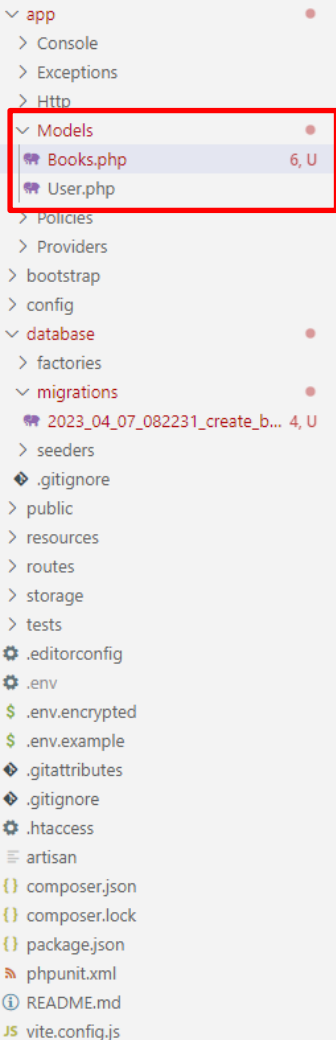
id: int
title: varchar(50)
author: varchar(50)
synopsis: int
story: text
rate: float
image_cover: text
is_active: tinyint
created_at: timestamp
updated_at: timestamp

Eloquent ORM



Eloquent ORM

Laravel includes Eloquent, an **object-relational mapper (ORM)** that makes it enjoyable to interact with your database. When using Eloquent, each database table has a corresponding "Model" that is used to interact with that table. In addition to retrieving records from the database table, Eloquent models allow you to insert, update, and delete records from the table as well.



```
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Support\Facades\DB;
8
9 class Books extends Model
10 {
11     use HasFactory;
12     protected $table = 'books';
13
14     protected $fillable = [
15         'id', 'title', 'author', 'synopsis', 'story', 'rate', 'image_cover', 'is_active'
16     ];
17
18     public function storedData($data){
19         $results = Books::create($data);
20         return $results;
21     }
22
23     public function updatedData($data){
24         $isExist = $this->getByCondition(array('id'=>$data['id']))->first();
25         if(!empty($isExist)){
26             unset($data['_token']);
27             $results = DB::table($this->table)->where(array('id'=>$data['id']))->update($data);
28             return $results;
29         }else{
30             return null;
31         }
32     }
33
34     public function getByCondition($condition){
35         $results = DB::table($this->table)->where($condition);
36         return $results;
37     }
38
39     public function removeByCondition($condition){
40         $results = Books::where($condition)->delete();
41         return $results;
42     }
43
44 }
```

Thanks