Pemograman Web

2022/2023 Genap

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

Kampus Merdeka
INDONESIA JAYA

INSTITUT BISNIS DAN INFORMATIKA KESATUAN
KESATUAN
BOGOR

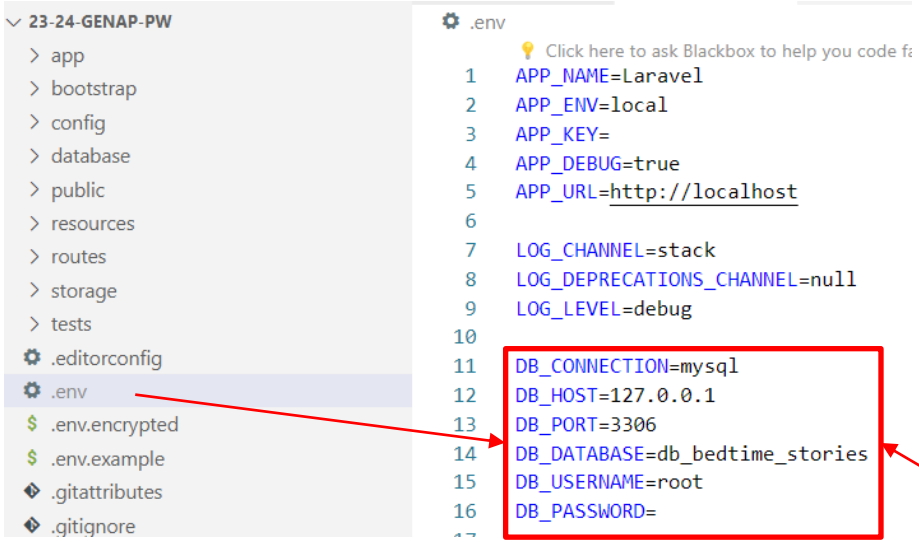
01

Website Databases

Migration and Eloquent ORM

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.

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.

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:

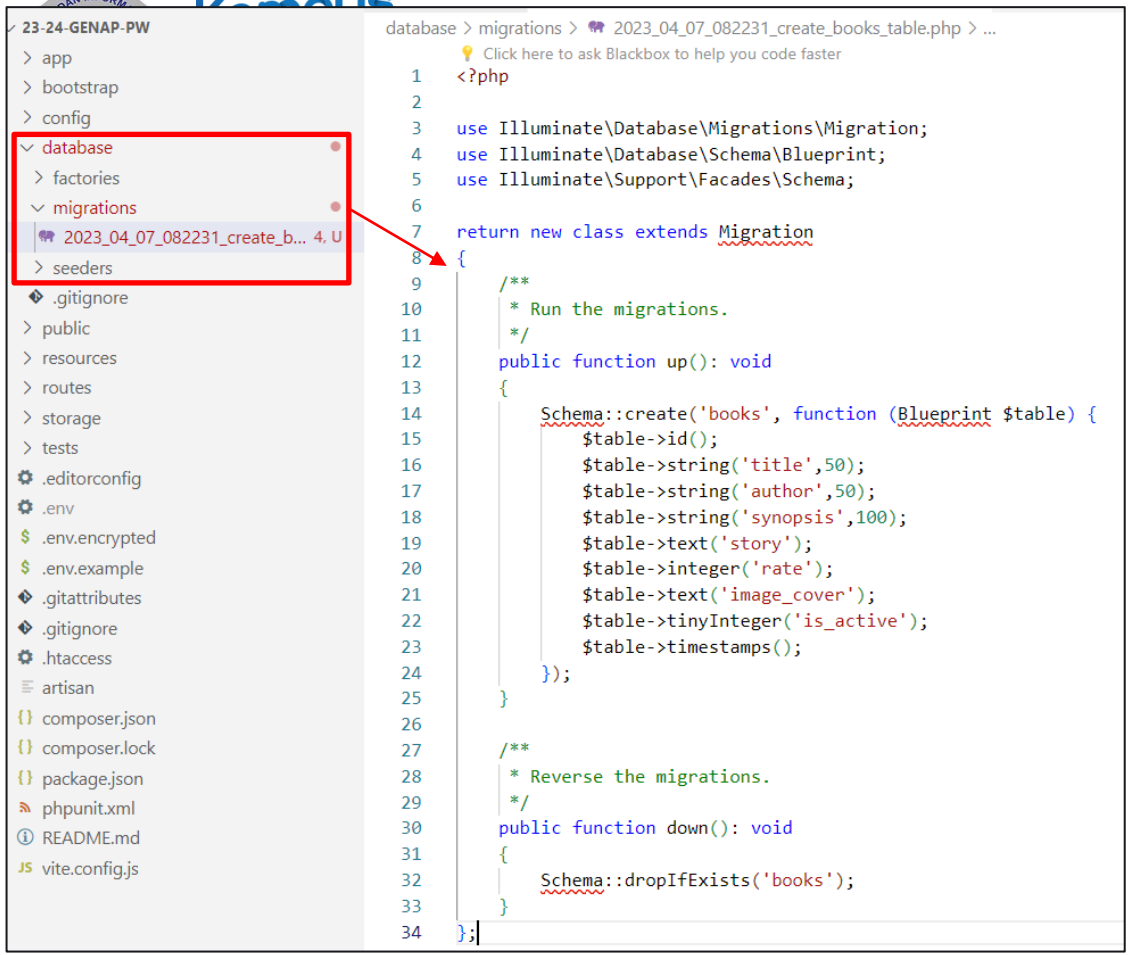
Syntax:

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

Sample:

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

https://laravel.com/docs/11.x/migrations



Type Data:

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

https://laravel.com/docs/11.x/migrations

Running Migrations

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

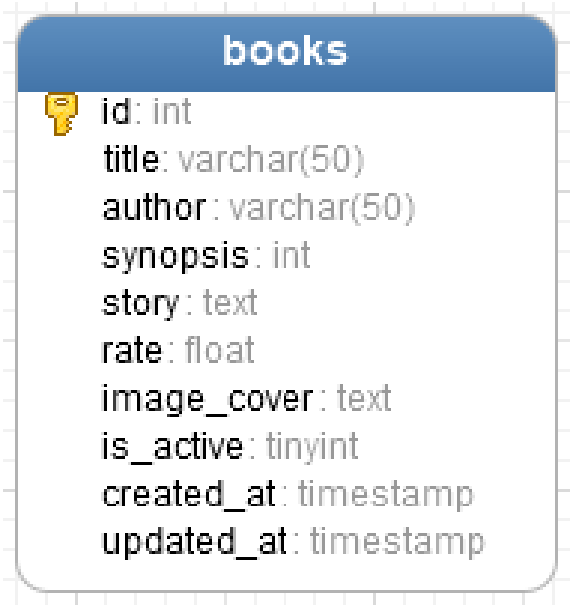
```
php artisan migrate
```

#Rolling Back Migrations

```
php artisan migrate:rollback
```

```
php artisan migrate:reset
```

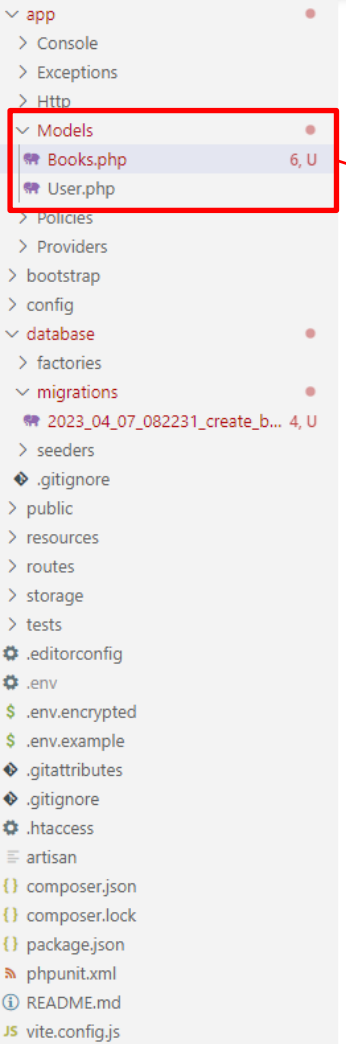
db_bedtime_stories



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.




```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class Books extends Model
{
    use HasFactory;
    protected $table = 'books';

    protected $fillable = [
        'id', 'title', 'author','synopsis','story','rate','image_cover','is_active'
    ];

    public function storedData($data){
        $results = Books::create($data);
        return $results;
    }

    public function updatedData($data){
        $isExist = $this->getByCondition(array('id'=>$data['id']))->first();
        if(!empty($isExist)){
            unset($data['_token']);
            $results = DB::table($this->table)->where(array('id'=>$data['id']))->update($data);
            return $results;
        }else{
            return null;
        }
    }

    public function getByCondition($condition){
        $results = DB::table($this->table)->where($condition);
        return $results;
    }

    public function removeByCondition($condition){
        $results = Books::where($condition)->delete();
        return $results;
    }

}
```

Thanks