API

+ Xampp

+ Laragon

+ Docker

```
// php artisan serve --host=127.0.0.1 --port=8089 // localhost
// php artisan serve --host=192.168.1.120 --port=8089 // ip
```

Source tham khao :

https://github.com/NguyenVanManh-AI/laravel-azure

https://github.com/NguyenVanManh-AI/linebot_be

https://laravel.com/docs/9.x

https://laracasts.com/series

https://www.vuemastery.com/courses/

- $ composer create-project --prefer-dist laravel/laravel:^9.0 traning_laravel9

Search ChatGPT Laravel : https://chatgpt.com/share/7548b394-3bf5-4802-bb3f-fd8769766205

- **Php artisan optimize:clear**
- **Php artisan optimize**

Dưới đây là toàn bộ các câu lệnh Artisan trong Laravel, được liệt kê theo từng nhóm chức năng chính:

Câu lệnh về Ứng dụng

- `artisan about`: Hiển thị thông tin về ứng dụng

- `artisan down`: Đưa ứng dụng vào chế độ bảo trì

- `artisan env`: Hiển thị môi trường hiện tại

- `artisan help`: Hiển thị trợ giúp cho một lệnh

- `artisan inspire`: Hiển thị một câu trích dẫn đầy cảm hứng

- `artisan list`: Liệt kê các lệnh có sẵn

- `artisan migrate`: Thực hiện các migration chưa thực hiện

- `artisan migrate:fresh`: Xóa và chạy lại tất cả các migration
- `artisan migrate:install`: Tạo bảng migration trong cơ sở dữ liệu
- `artisan migrate:refresh`: Xóa và chạy lại tất cả các migration mà không làm mất dữ liệu
- `artisan migrate:reset`: Xóa toàn bộ các migration đã thực hiện
- `artisan migrate:rollback`: Rollback migration gần nhất
- `artisan migrate:status`: Hiển thị trạng thái của các migration
- `artisan optimize`: Tối ưu hóa bộ nhớ đệm của framework
- `artisan serve`: Khởi động máy chủ phát triển
- `artisan test`: Chạy các bài kiểm thử
- `artisan up`: Đưa ứng dụng ra khỏi chế độ bảo trì

Câu lệnh về Auth

- `artisan make:auth`: Tạo scaffoldings cơ bản cho chức năng xác thực

Câu lệnh về Bộ nhớ đệm (Cache)

- `artisan cache:clear`: Xóa toàn bộ bộ nhớ đệm
- `artisan cache:forget`: Xóa một mục trong bộ nhớ đệm
- `artisan cache:table`: Tạo migration cho bảng bộ nhớ đệm

Câu lệnh về Cấu hình

- `artisan config:cache`: Tạo bộ nhớ đệm cho file cấu hình
- `artisan config:clear`: Xóa bộ nhớ đệm của file cấu hình

Câu lệnh về Cơ sở dữ liệu (DB)

- `artisan db:seed`: Chạy các seeders

- `artisan db:wipe`: Xóa toàn bộ cơ sở dữ liệu

Câu lệnh về Event

- `artisan event:cache`: Tạo bộ nhớ đệm cho các sự kiện và listeners
- `artisan event:clear`: Xóa bộ nhớ đệm của các sự kiện và listeners
- `artisan event:generate`: Tạo các lớp event và listener
- `artisan event:list`: Liệt kê các sự kiện và listeners

Câu lệnh về Job

- `artisan queue:failed`: Hiển thị danh sách các job đã thất bại
- `artisan queue:failed-table`: Tạo migration cho bảng job thất bại
- `artisan queue:flush`: Xóa toàn bộ job thất bại
- `artisan queue:forget`: Xóa một job thất bại
- `artisan queue:listen`: Khởi động worker để lắng nghe queue
- `artisan queue:restart`: Khởi động lại tất cả các queue worker
- `artisan queue:retry`: Thử lại một job thất bại
- `artisan queue:table`: Tạo migration cho bảng queue

Câu lệnh về Key

- `artisan key:generate`: Tạo khóa ứng dụng

Câu lệnh về Log

- `artisan log:clear`: Xóa tất cả các file log (cần phải cài thêm gói `ignited/laravel-4-generators`)

Câu lệnh về Make

- `artisan make:channel`: Tạo một lớp channel mới
- `artisan make:command`: Tạo một lệnh Artisan mới
- `artisan make:controller`: Tạo một controller mới
- `artisan make:event`: Tạo một event mới
- `artisan make:exception`: Tạo một exception mới
- `artisan make:factory`: Tạo một factory mới
- `artisan make:job`: Tạo một job mới
- `artisan make:listener`: Tạo một listener mới
- `artisan make:mail`: Tạo một mailable mới
- `artisan make:middleware`: Tạo một middleware mới
- `artisan make:migration`: Tạo một migration mới
- `artisan make:model`: Tạo một model mới
- `artisan make:notification`: Tạo một notification mới
- `artisan make:observer`: Tạo một observer mới
- `artisan make:policy`: Tạo một policy mới
- `artisan make:provider`: Tạo một service provider mới
- `artisan make:request`: Tạo một form request mới
- `artisan make:resource`: Tạo một resource mới
- `artisan make:rule`: Tạo một validation rule mới
- `artisan make:seeder`: Tạo một seeder mới
- `artisan make:test`: Tạo một test mới

Câu lệnh về Queue

- `artisan queue:work`: Khởi động worker để xử lý queue

Câu lệnh về Route

- `artisan route:cache`: Tạo bộ nhớ đệm cho route

- `artisan route:clear`: Xóa bộ nhớ đệm của route

- `artisan route:list`: Liệt kê tất cả các route

Câu lệnh về Storage

- `artisan storage:link`: Tạo symbolic link từ "public/storage" tới "storage/app/public"

Câu lệnh về View

- `artisan view:cache`: Tạo bộ nhớ đệm cho view

- `artisan view:clear`: Xóa bộ nhớ đệm của view

Bạn có thể chạy `php artisan list` trong terminal của bạn để xem tất cả các lệnh Artisan có sẵn cho phiên bản Laravel mà bạn đang sử dụng.

Traning 1

https://packagist.org/packages/nesbot/carbon#3.0.2

https://www.npmjs.com/package/rich-text-editor

table user

user -> article

seeder => sinh dữ liệu

Trong Laravel, khi bạn tạo một bảng cơ sở dữ liệu bằng cách sử dụng migration, bạn có thể định nghĩa các loại dữ liệu khác nhau cho các cột trong bảng. Dưới đây là tất cả các loại dữ liệu mà bạn có thể sử dụng trong migrations của Laravel:

Loại dữ liệu chính

1. **`bigIncrements('id')`**: Tự động tăng, khóa chính (unsigned BIGINT).
2. **`bigInteger('votes')`**: Số nguyên lớn (BIGINT).
3. **`binary('data')`**: Dữ liệu nhị phân (BLOB).
4. **`boolean('confirmed')`**: Boolean (TINYINT).
5. **`char('name', 4)`**: Chuỗi ký tự cố định (CHAR) với độ dài tùy chỉnh.
6. **`date('created_at')`**: Ngày (DATE).
7. **`dateTime('created_at')`**: Ngày và giờ (DATETIME).
8. **`dateTimeTz('created_at')`**: Ngày và giờ với múi giờ (DATETIME với múi giờ).
9. **`decimal('amount', 8, 2)`**: Số thập phân (DECIMAL) với độ chính xác và tỉ lệ tùy chỉnh.
10. **`double('amount', 8, 2)`**: Số thập phân (DOUBLE) với độ chính xác và tỉ lệ tùy chỉnh.
11. **`enum('level', ['easy', 'hard'])`**: Giá trị enum.
12. **`float('amount', 8, 2)`**: Số thập phân (FLOAT) với độ chính xác và tỉ lệ tùy chỉnh.
13. **`foreignId('user_id')`**: Khóa ngoại (unsigned BIGINT).
14. **`geometry('location')`**: Dữ liệu hình học.
15. **`geometryCollection('shapes')`**: Bộ sưu tập dữ liệu hình học.
16. **`id()`**: Tự động tăng, khóa chính (unsigned BIGINT) (alias của bigIncrements).
17. **`increments('id')`**: Tự động tăng, khóa chính (unsigned INT).
18. **`integer('votes')`**: Số nguyên (INT).
19. **`ipAddress('visitor')`**: Địa chỉ IP (VARCHAR).
20. **`json('options')`**: Dữ liệu JSON (TEXT).
21. **`jsonb('options')`**: Dữ liệu JSONB (JSON binary).
22. **`lineString('routes')`**: Chuỗi đường (LineString).
23. **`longText('description')`**: Văn bản dài (LONGTEXT).
24. **`macAddress('device')`**: Địa chỉ MAC (VARCHAR).
25. **`mediumIncrements('id')`**: Tự động tăng, khóa chính (unsigned MEDIUMINT).
26. **`mediumInteger('votes')`**: Số nguyên trung bình (MEDIUMINT).
27. **`mediumText('description')`**: Văn bản trung bình (MEDIUMTEXT).
28. **`morphs('taggable')`**: Khóa ngoại morph (ID và type).
29. **`multiLineString('routes')`**: Chuỗi nhiều đường (MultiLineString).
30. **`multiPoint('positions')`**: Nhiều điểm (MultiPoint).
31. **`multiPolygon('areas')`**: Nhiều đa giác (MultiPolygon).
32. **`nullableMorphs('taggable')`**: Khóa ngoại morph (ID và type), có thể null.
33. **`nullableTimestamps()`**: Timestamps (created_at, updated_at), có thể null.
34. **`point('position')`**: Điểm (Point).
35. **`polygon('area')`**: Đa giác (Polygon).
36. **`rememberToken()`**: Token remember (VARCHAR).
37. **`set('flavors', ['strawberry', 'vanilla'])`**: Tập hợp giá trị (SET).
38. **`smallIncrements('id')`**: Tự động tăng, khóa chính (unsigned SMALLINT).
39. **`smallInteger('votes')`**: Số nguyên nhỏ (SMALLINT).
40. **`softDeletes()`**: Soft deletes (deleted_at).
41. **`softDeletesTz()`**: Soft deletes với múi giờ (deleted_at với múi giờ).
42. **`string('name', 100)`**: Chuỗi ký tự (VARCHAR) với độ dài tùy chỉnh.
43. **`text('description')`**: Văn bản (TEXT).
44. **`time('sunrise')`**: Thời gian (TIME).
45. **`timeTz('sunrise')`**: Thời gian với múi giờ (TIME với múi giờ).

46. `timestamp('added_on')`: Dấu thời gian (TIMESTAMP).
47. `timestampTz('added_on')`: Dấu thời gian với múi giờ (TIMESTAMP với múi giờ).
48. `timestamps()`: Timestamps (created_at, updated_at).
49. `timestampsTz()`: Timestamps với múi giờ (created_at, updated_at với múi giờ).
50. `tinyIncrements('id')`: Tự động tăng, khóa chính (unsigned TINYINT).
51. `tinyInteger('votes')`: Số nguyên rất nhỏ (TINYINT).
52. `unsignedBigInteger('votes')`: Số nguyên không dấu lớn (unsigned BIGINT).
53. `unsignedDecimal('amount', 8, 2)`: Số thập phân không dấu (unsigned DECIMAL) với độ chính xác và tỉ lệ tùy chỉnh.
54. `unsignedInteger('votes')`: Số nguyên không dấu (unsigned INT).
55. `unsignedMediumInteger('votes')`: Số nguyên trung bình không dấu (unsigned MEDIUMINT).
56. `unsignedSmallInteger('votes')`: Số nguyên nhỏ không dấu (unsigned SMALLINT).
57. `unsignedTinyInteger('votes')`: Số nguyên rất nhỏ không dấu (unsigned TINYINT).
58. `uuid('id')`: UUID (CHAR(36)).
59. `year('birth_year')`: Năm (YEAR).

Các phương thức tiện ích

- `after('column')`: Đặt cột mới sau cột đã chỉ định.
- `before('column')`: Đặt cột mới trước cột đã chỉ định.
- `first()`: Đặt cột mới ở vị trí đầu tiên.
- `nullable()`: Cho phép cột nhận giá trị null.
- `default($value)`: Đặt giá trị mặc định cho cột.
- `change()`: Thay đổi cột hiện tại.
- `index()`: Tạo chỉ mục cho cột.
- `unique()`: Tạo chỉ mục duy nhất cho cột.
- `primary()`: Đặt cột làm khóa chính.
- `foreign()`: Đặt cột làm khóa ngoại.

Bạn có thể kết hợp các loại dữ liệu và các phương thức tiện ích này để tạo ra cấu trúc bảng cơ sở dữ liệu phù hợp với yêu cầu của ứng dụng.

Eloquent là ORM (Object-Relational Mapper) của Laravel, cho phép bạn làm việc với cơ sở dữ liệu một cách dễ dàng bằng cách sử dụng các đối tượng PHP. Dưới đây là danh sách các phương thức và tính năng chính của Eloquent:

Khởi tạo và Truy xuất Dữ liệu

- `all()`: Truy xuất tất cả các bản ghi.
- `find($id): Tìm bản ghi theo ID.`
- `findOrFail($id)`: Tìm bản ghi theo ID, ném ngoại lệ nếu không tìm thấy.
- `findOrNew`
- firstOrCreate
- `first()`: Lấy bản ghi đầu tiên.

- `firstOrFail()`: Lấy bản ghi đầu tiên, ném ngoại lệ nếu không tìm thấy.
- `get()`: Lấy tập hợp các bản ghi.
- `pluck('column')`: Lấy danh sách giá trị của một cột.
- `value('column')`: Lấy giá trị của một cột từ bản ghi đầu tiên.
- `where('column', 'operator', 'value')`: Điều kiện truy vấn.
- `orWhere('column', 'operator', 'value')`: Điều kiện OR.
- `whereIn('column', ['value1', 'value2'])`: Điều kiện IN.
- `whereNotIn('column', ['value1', 'value2'])`: Điều kiện NOT IN.
- `whereNull('column')`: Điều kiện IS NULL.
- `whereNotNull('column')`: Điều kiện IS NOT NULL.
- `whereBetween('column', [$value1, $value2])`: Điều kiện BETWEEN.
- `whereNotBetween('column', [$value1, $value2])`: Điều kiện NOT BETWEEN.
- `with('relation')`: Eager load mối quan hệ.
- `withCount('relation')`: Eager load và đếm mối quan hệ.
- `has('relation')`: Điều kiện tồn tại mối quan hệ.
- `doesntHave('relation')`: Điều kiện không tồn tại mối quan hệ.

Tạo, Cập nhật và Xóa Dữ liệu

- `create(['column' => 'value'])`: Tạo bản ghi mới.
- `update(['column' => 'value'])`: Cập nhật bản ghi.
- `save()`: Lưu bản ghi hiện tại.
    - `$user = User();`
    - `$user->id = 1`
    - `Code something...`
    - `$user->name = 'NVM'`
    - `....`
    - `$user->save()`
- `delete()`: Xóa bản ghi hiện tại.
    - `$user = User::find(2);`
    - `$user->delete();` => xóa user có id = 2
- `destroy($id)`: Xóa bản ghi theo ID.
- `truncate()`: Xóa tất cả các bản ghi.

Truy vấn Nâng cao

- `orderBy('column', 'direction')`: Sắp xếp kết quả. (DESC, ASC)
- `groupBy('column')`: Nhóm kết quả theo cột.
- `having('column', 'operator', 'value')`: Điều kiện HAVING.
- `skip($value)`: Bỏ qua số lượng bản ghi.
- `take($value)`: Giới hạn số lượng bản ghi.
- `offset($value)`: Bắt đầu từ vị trí bản ghi.
- `limit($value)`: Giới hạn số lượng bản ghi.

Quan hệ (Relationships)

- `hasOne('App\RelatedModel')`: Mối quan hệ một-một.
- `belongsTo('App\RelatedModel')`: Mối quan hệ thuộc về.

- **`hasMany('App\RelatedModel')`**: Mối quan hệ một-nhiều.
- **`belongsToMany('App\RelatedModel')`**: Mối quan hệ nhiều-nhiều.
- **`morphTo()`**: Mối quan hệ đa hình.
- **`morphOne('App\RelatedModel')`**: Mối quan hệ đa hình một-một.
- **`morphMany('App\RelatedModel')`**: Mối quan hệ đa hình một-nhiều.
- **`morphToMany('App\RelatedModel')`**: Mối quan hệ đa hình nhiều-nhiều.

Eager Loading

- **`load('relation')`**: Lazy eager load mối quan hệ.
- **`with('relation.subRelation')`**: Eager load mối quan hệ và các mối quan hệ con.

Tiện ích Khác

- **`count()`**: Đếm số lượng bản ghi.
- **`max('column')`**: Lấy giá trị lớn nhất của cột.
- **`min('column')`**: Lấy giá trị nhỏ nhất của cột.
- **`avg('column')`**: Lấy giá trị trung bình của cột.
- **`sum('column')`**: Lấy tổng giá trị của cột.
- **`exists()`**: Kiểm tra sự tồn tại của bản ghi.
- **`touch()`**: Cập nhật timestamps.

Soft Deletes

- **`withTrashed()`**: Bao gồm các bản ghi đã bị xóa mềm trong kết quả.
- **`onlyTrashed()`**: Chỉ lấy các bản ghi đã bị xóa mềm.
- **`restore()`**: Khôi phục bản ghi đã bị xóa mềm.
- **`forceDelete()`**: Xóa vĩnh viễn bản ghi.

Tùy chỉnh Query Builder

- **`select('column')`**: Chọn các cột cụ thể.
- **`addSelect('column')`**: Thêm các cột vào lựa chọn.
- **`distinct()`**: Lấy các bản ghi duy nhất.
- **`raw('expression')`**: Sử dụng biểu thức SQL thô.

Dưới đây là một ví dụ về cách sử dụng Eloquent để truy vấn và thao tác với dữ liệu:

```
php
Copy code
use App\Models\User;
use App\Models\Post;

// Truy xuất tất cả người dùng
$users = User::all();

// Tìm người dùng theo ID
$user = User::find(1);
```

```
// Tạo người dùng mới
User::create(['name' => 'John Doe', 'email' => 'john@example.com']);

// Cập nhật người dùng
$user->update(['name' => 'Jane Doe']);

// Xóa người dùng
$user->delete();

// Quan hệ một-nhiều
$posts = User::find(1)->posts;

// Eager load quan hệ
$users = User::with('posts')->get();
```

Eloquent cung cấp nhiều tính năng và phương thức mạnh mẽ giúp làm việc với cơ sở dữ liệu dễ dàng và hiệu quả. Bạn có thể kết hợp các phương thức này để xây dựng các truy vấn phức tạp và quản lý dữ liệu một cách hiệu quả.

Trong Eloquent của Laravel, thứ tự các phương thức được gọi có thể ảnh hưởng đến kết quả của truy vấn. Dưới đây là một số hướng dẫn về thứ tự gọi các phương thức Eloquent:

1. **Khởi tạo truy vấn:** Bắt đầu với việc khởi tạo đối tượng truy vấn trên model.
2. **Điều kiện truy vấn:** Áp dụng các điều kiện truy vấn (`where`, `orWhere`, `whereIn`, v.v.).
3. **Quan hệ:** Eager load các mối quan hệ nếu cần (`with`, `withCount`).
4. **Sắp xếp và nhóm:** Áp dụng các điều kiện sắp xếp và nhóm (`orderBy`, `groupBy`).
5. **Giới hạn và offset:** Áp dụng các điều kiện giới hạn và offset (`limit`, `offset`).
6. **Lấy dữ liệu:** Kết thúc với các phương thức lấy dữ liệu (`get`, `first`, `pluck`, v.v.).

Ví dụ:

```
php
Copy code
use App\Models\User;

// Bắt đầu với đối tượng truy vấn
$query = User::query();

// Áp dụng các điều kiện truy vấn
$query->where('status', 'active')
      ->where('age', '>', 18);

// Eager load các mối quan hệ
$query->with('posts', 'comments');

// Áp dụng sắp xếp và nhóm
$query->orderBy('name', 'asc')
      ->groupBy('country');

// Áp dụng giới hạn và offset
$query->limit(10)
      ->offset(5);
```

```
// Lấy dữ liệu
$users = $query->get();
```

Thứ tự các hàm chi tiết hơn

1. **Khởi tạo truy vấn:**
    - `Model::query()`
    - `Model::where()`
2. **Điều kiện truy vấn:**
    - `where()`
    - `orWhere()`
    - `whereIn()`
    - `whereNotIn()`
    - `whereNull()`
    - `whereNotNull()`
    - `whereBetween()`
    - `whereNotBetween()`
    - `whereDate()`
    - `whereMonth()`
    - `whereDay()`
    - `whereYear()`
    - `whereTime()`
    - `whereColumn()`
3. **Quan hệ:**
    - `with()`
    - `withCount()`
    - `has()`
    - `doesntHave()`
    - `whereHas()`
    - `whereDoesntHave()`
    - `withTrashed()`
    - `onlyTrashed()`
    - `withoutTrashed()`
4. **Sắp xếp và nhóm:**
    - `orderBy()`
    - `orderByDesc()`
    - `latest()`
    - `oldest()`
    - `inRandomOrder()`
    - `groupBy()`
    - `having()`
    - `havingRaw()`
    - `orderByRaw()`
    - `groupByRaw()`
5. **Giới hạn và offset:**
    - `limit()`
    - `offset()`
    - `skip()`
    - `take()`
6. **Lấy dữ liệu:**
    - `get()`
    - `first()`
    - `firstOrFail()`
    - `find()`
    - `findOrFail()`
    - `pluck()`

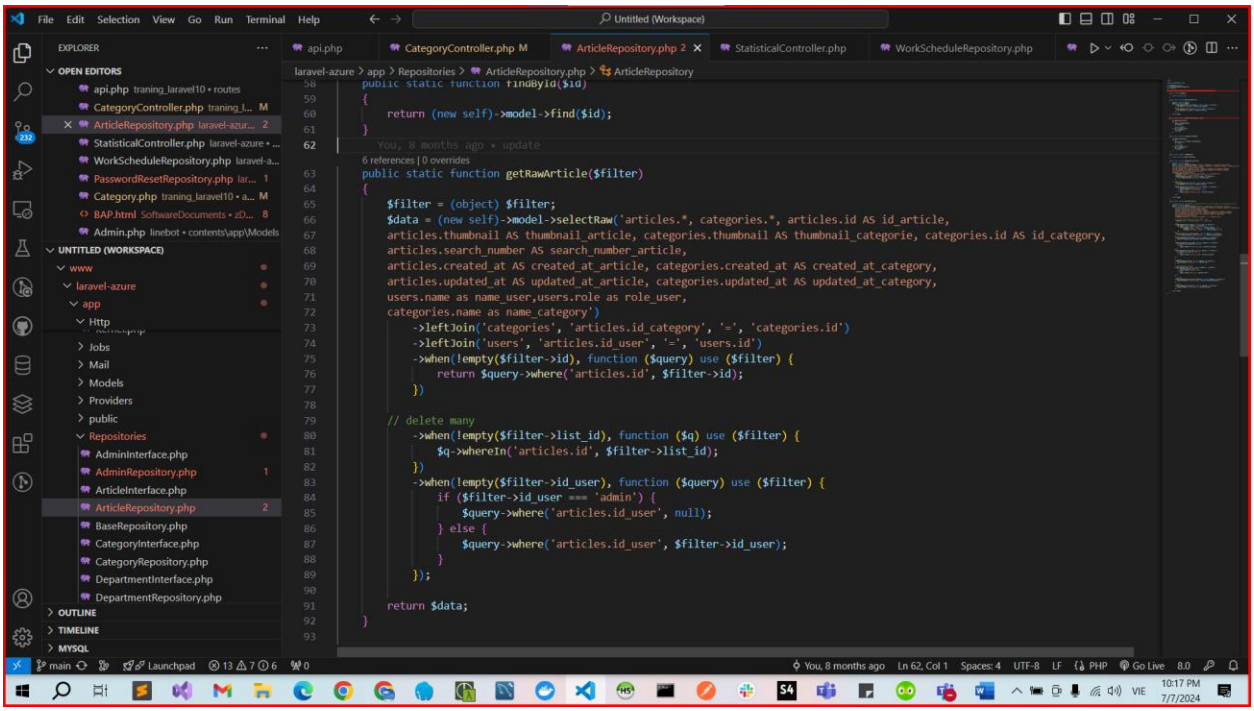
- value()
- count()
- max()
- min()
- avg()
- sum()
- exists()
- doesntExist()

Ví dụ phức tạp hơn:

```
php
Copy code
use App\Models\Post;

// Bắt đầu với đối tượng truy vấn
$query = Post::query();

// Áp dụng các điều kiện truy vấn
$query->where('published', true)
      ->where('views', '>', 100);

// Eager load các mối quan hệ và đếm
$query->with(['author', 'comments'])
      ->withCount('comments');

// Áp dụng sắp xếp và nhóm
$query->orderBy('created_at', 'desc')
      ->groupBy('author_id');

// Áp dụng giới hạn và offset
$query->limit(10)
      ->offset(0);

// Lấy dữ liệu
$posts = $query->get();
```

Trong ví dụ này:

- Bắt đầu bằng cách khởi tạo truy vấn trên model `Post`.
- Áp dụng các điều kiện truy vấn (`where`).
- Eager load các mối quan hệ (`with`, `withCount`).
- Áp dụng điều kiện sắp xếp và nhóm (`orderBy`, `groupBy`).
- Áp dụng giới hạn và offset (`limit`, `offset`).
- Kết thúc bằng việc lấy dữ liệu (`get`).


File Edit Selection View Go Run Terminal Help
Untitled (Workspace)
EXPLORER
OPEN EDITORS
api.php traning_laravel10 • routes
CategoryController.php traning_l... M
ArticleRepository.php laravel-azur... 2
StatisticalController.php laravel-azure • ...
WorkScheduleRepository.php laravel-a...
PasswordResetRepository.php lar... 1
Category.php traning_laravel10 • a... M
BAP.html SoftwareDocuments • zD... 8
Admin.php linebot • contents\app\Models
UNTITLED (WORKSPACE)
www
laravel-azure
app
Http
Jobs
Mail
Models
Providers
public
Repositories
AdminInterface.php
AdminRepository.php 1
ArticleInterface.php
ArticleRepository.php 2
BaseRepository.php
CategoryInterface.php
CategoryRepository.php
DepartmentInterface.php
DepartmentRepository.php
OUTLINE
TIMELINE
MYSQL
api.php
CategoryController.php M
ArticleRepository.php 2
StatisticalController.php
WorkScheduleRepository.php
laravel-azure > app > Repositories > ArticleRepository.php > ArticleRepository
public static function findById($id)
{
return (new self)->model->find($id);
}
You, 8 months ago • update
6 references | 0 overrides
public static function getRawArticle($filter)
{
$filter = (object) $filter;
$data = (new self)->model->selectRaw('articles.*, categories.*, articles.id AS id_article,
articles.thumbnail AS thumbnail_article, categories.thumbnail AS thumbnail_categorie, categories.id AS id_category,
articles.search_number AS search_number_article,
articles.created_at AS created_at_article, categories.created_at AS created_at_category,
articles.updated_at AS updated_at_article, categories.updated_at AS updated_at_category,
users.name as name_user,users.role as role_user,
categories.name as name_category')
->leftJoin('categories', 'articles.id_category', '=', 'categories.id')
->leftJoin('users', 'articles.id_user', '=', 'users.id')
->when(!empty($filter->id), function ($query) use ($filter) {
return $query->where('articles.id', $filter->id);
})
// delete many
->when(!empty($filter->list_id), function ($q) use ($filter) {
$q->whereIn('articles.id', $filter->list_id);
})
->when(!empty($filter->id_user), function ($query) use ($filter) {
if ($filter->id_user === 'admin') {
$query->where('articles.id_user', null);
} else {
$query->where('articles.id_user', $filter->id_user);
}
});
return $data;
}
main Launchpad 13 7 6 0
You, 8 months ago Ln 62, Col 1 Spaces: 4 UTF-8 LF PHP Go Live 8.0
VIE 10:17 PM 7/7/2024