

Представляємо Laravel Scout

🕒 09.06.2019 👤 Cinex 📁 Веб-розробка 💬 0



ПОШУК

ПОЗНАЧКИ

[APACHE](#)[AUTORUN](#)[BASH](#)[BITCOIN](#)[BLOCKCHAIN](#)[COMPOSER](#)[CRYPTOCURRENCY](#)[CSS](#)[DATABASE](#)[DJANGO](#)[DOCKER](#)[ELOQUENT](#)[ENGLISH](#)[ENUM](#)[GAMES](#)[GIMP](#)[GIT](#)[HTML](#)[JAVA](#)[JAVASCRIPT](#)[JQUERY](#)[LARAVEL](#)[MERCURIAL](#)[PARSING](#)[PHOTOSHOP](#)[PHP](#)[PHPSTORM](#)

Інструменти пошуку Elasticsearch і Algolia набули неабиякої популярності в співтоваристві Laravel за останні кілька років як потужні інструменти для індексації і пошуку ваших даних. Бен Корлетт проробив фантастичну роботу, представивши [ElasticSearch на Laracon Eu 2014](#), і я написав [запит на Laravel.com, щоб ввести засновану на Elasticsearch індексацію для документів](#) в 2015 році. Але до того, як мій PR був об'єднаний, хлопці з Algolia взяли і відновили його. замість цього використовувати Algolia (швидше і з кращим призначенням для користувача інтерфейсом!), і це те, що ви побачите сьогодні, якщо будете шукати в документах Laravel.

Якщо ви подивитесь на мій запит на отримання або його [запит](#), ви побачите, що завдання інтегрувати повнотекстовий пошук в ваш сайт - завдання не з легких. З тих пір Algolia випустила безкоштовний продукт під назвою Algolia DocSearch, який дозволяє легко [додавати пошуковий віджет Algolia](#) на сторінки документації. Але для всього іншого ви все ще застрягли, написавши інтеграцію самостійно, тобто до сих пір.

Представляємо Laravel Scout

Scout - це засноване на драйверах рішення для повнотекстового пошуку Eloquent. Scout дозволяє легко індексувати і шукати вміст ваших моделей Eloquent; в даний час він працює з Algolia і Elasticsearch, але Тейлор попросив спільноти внести свій вклад в інші служби повнотекстового пошуку.

Scout - це окремий пакет Laravel, такий як Cashier, який вам потрібно використовувати в Composer. Ми будемо додавати в наші моделі риси, які вказують Scout, що він повинен прослуховувати події, що виникають при зміні примірників цих моделей, і оновлювати пошуковий індекс у відповідь.

Погляньте на цей синтаксис для повнотекстового пошуку, для пошуку будь-якого огляду зі словом Llew в ньому:

[PLAYONLINUX](#)[PLUGINS](#)[PROBLEM SOLVING](#)[PYTHON](#)[REGEXP](#)[SEO](#)[SOFTWARE ENGINEERING](#)[SPRING](#)[SQL](#)[TELEGRAM](#)[TIPS AND TRICKS](#)[TIPS AND TRICKS](#)[UNIT TESTING](#)[WORDPRESS](#)[ВІРШІ](#)[ШПАРГАЛКИ](#)

ОСТАННІ НОТАТКИ

20 Laravel Eloquent порад і трюків

15.12.2019

Як зробити скріншот сайту по URL на PHP

04.08.2019

Docker Cheat Sheet

05.07.2019

Гарячі клавіші Ubuntu Linux

01.07.2019

Git happens! 6 типових помилок Git і як їх виправити

22.06.2019

НЕДАВНІ КОМЕНТАРІ

```
1 Review::search('Llew')->get();
2 Review::search('Llew')->paginate(20);
3 Review::search('Llew')->where('account_id', 2)->get();
```

Все це з дуже маленькою конфігурацією. Це прекрасна річ.

Встановлення Scout

Спочатку встановіть пакет (як тільки він з'явиться, і в додатку Laravel 5.3):

```
1 composer require laravel/scout
```

Потім додайте Provider послуг Scout (Laravel\Scout\ScoutServiceProvider :: class) в розділ Providers в config/app.php.

Ми хочемо налаштувати нашу конфігурацію скаутів. Запустіть php artisan vendor:publish і вставте свої облікові дані Algolia в config/scout.php.

Нарешті, якщо ви використовуєте Algolia, встановіть Algolia SDK:

```
1 composer require algolia/algoliasearch-client-php
```

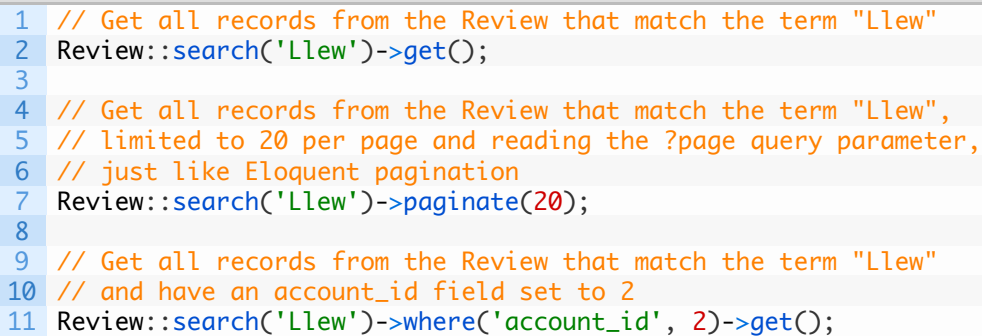
Маркування вашої моделі для індексації

Тепер перейдіть до вашої моделі (для цього прикладу ми будемо використовувати Review, для рецензування книг). Імпорт трейт Laravel\Scout\Searchable. Ви можете визначити, які властивості доступні для пошуку, за допомогою методу toSearchableArray() (за замовчуванням це дзеркальне відображення toArray()), а також визначити ім'я індексу моделі за допомогою методу searchableAs() (за замовчуванням це ім'я таблиці).

Як тільки ми це зробимо, ви можете перевірити свою сторінку індексу Алголь на їх веб-сайті; Коли ви додаєте, оновлюєте або видаляєте записи, ви побачите оновлення індексу Algolia. Просто так.

Пошук в вашому індексі

Ми вже подивилися на це, але ось оновлення про те, як шукати:



```
1 // Get all records from the Review that match the term "Llew"
2 Review::search('Llew')->get();
3
4 // Get all records from the Review that match the term "Llew",
5 // limited to 20 per page and reading the ?page query parameter,
6 // just like Eloquent pagination
7 Review::search('Llew')->paginate(20);
8
9 // Get all records from the Review that match the term "Llew"
10 // and have an account_id field set to 2
11 Review::search('Llew')->where('account_id', 2)->get();
```

Що повертається з цих пошуків? Колекція моделей Eloquent, відновлена з вашої бази даних. Ідентифікатори зберігаються в Algolia, яка повертає список співпадаючих ідентифікаторів, а потім Scout витягує записи бази даних для них і повертає їх як об'єкти Eloquent.

У вас немає повного доступу до складності SQL, де команди, але він обробляє міцну базову структуру для перевірок порівняння, як ви можете бачити в прикладах коду вище.

Черги

Ви, ймовірно, можете здогадатися, що ми зараз відправляємо HTTP-запити в Algolia на кожен запит, який змінює будь-які записи в базі даних. Це може привести до дуже швидкого уповільнення роботи, тому ви можете захотіти поставити в чергу ці операції, що, на щастя, просто.

У config/scout.php встановіть для черги значення true, щоб ці оновлення були проіндексовані асинхронно. Зараз ми дивимося на «можливу послідовність»; ваші записи в базі даних будуть отримувати оновлення негайно, а поновлення ваших пошукових індексів будуть ставитися в чергу і оновлюватися так швидко, як дозволяє ваш працівник черзі.

Особливі випадки

Давайте розглянемо деякі особливі випадки.

Виконувати операції без індексації

Що якщо ви хочете виконати набір операцій і уникнути запуску індексації у відповідь? Просто помістіть їх в метод withoutSyncingToSearch() вашої моделі:

```
1 Review::withoutSyncingToSearch(function () {  
2     // make a bunch of reviews, e.g.  
3     factory(Review::class, 10)->create();  
4 });
```

Запустити індексацію вручну за допомогою коду

Припустимо, ви тепер готові виконувати індекси, тепер, коли якась масова операція була успішно виконана. Як?

Просто додайте searchable() в кінець будь-якого запиту Eloquent, і він буде індексувати всі записи, знайдені в цьому запиті.

```
1 Review::all()->searchable();
```

Ви також можете вибрати охоплення запиту тільки тими, які хочете проіндексувати, але варто відзначити, що при індексуванні вставлятимуться нові записи і оновлюватися старі записи,

тому непогано дозволити запускати деякі записи, які вже можуть бути проіндексовані.

Це також буде працювати на відносини:

```
1 $user->reviews()->searchable();
```

Ви також можете скасувати індексацію будь-яких записів з таким же типом ланцюжка запитів, але замість цього просто використовуючи `unsearchable()`:

```
1 Review::where('sucky', true)->unsearchable();
```

Запуск індексації вручну через CLI

Для цього є команда `artisan`.

```
1 php artisan scout:import App\\Review
```

Це розділить на частини всі моделі огляду і внесе їх у показчик.

Висновок

Це воно! Майже без роботи у вас тепер є повний повнотекстовий пошук на ваших моделях Eloquent.

Джерело



LARAVEL

PHP

« ПОПЕРЕДНІЙ

ДАЛІ »



Вивчаємо Eloquent: API Resources в
Laravel 5.5

Новий синтаксис JSON-стовпця
where() і update() в Laravel 5.3



ЗАЛИШТЕ ПЕРШИЙ КОМЕНТАР

Залишити коментар

Вашу адресу електронної пошти не буде опубліковано

Коментувати

Ім'я*

Email *

☐ Збережіть моє ім'я, електронну пошту у цьому веб-переглядачі наступного разу, коли я коментуватиму.

ОПУБЛІКУВАТИ КОМЕНТАР

