DATABASE: PAGINATION

1.1. Introduction

Trong các frameworks khác, việc phân trang có thể rất khó khăn. Trong Laravel, Laravel's paginator (phân trang) được tích hợp với query builder và Eloquent ORM và cung cấp phân trang thuận tiện, dễ sử dụng cho các bản ghi cơ sở dữ liệu.

Theo mặc định, HTML được tạo bởi bộ phân trang tương thích với Tailwind CSS framework; tuy nhiên, hỗ trợ phân trang Bootstrap cũng có sẵn.

1.2. Basic Usage

1.2.1. Paginating Query Builder Results

Có một số cách để phân trang các items. Đơn giản nhất là sử dụng phương thức paginate trên query builder hoặc Eloquent ORM. Phương thức paginate tự động đảm nhận việc đặt "limit" và "offset" của truy vấn dựa trên trang hiện tại đang được xem bởi người dùng. Theo mặc định, trang hiện tại được xác định bởi giá trị của đối số chuỗi truy vấn page trên HTTP request. Giá trị này được Laravel tự động xác định và chèn vào các liên kết do paginator tạo ra.

Cú pháp:

```
paginate($perPage, $columns, $pageName, $page);
```

Trong đó:

- $perPage là số lượng item sẽ lấy ra và hiển thị trên mỗi trang. Mặc định sẽ là 15 item / mỗi trang;
- $columns là những cột sẽ lấy ra trong Database. Mặc định sẽ lấy hết (SELETC *);
- $pageName là tên của query string sẽ chứa tham số page number. Mặc định $pageName = 'page';
- $page là item muốn lấy ra là trang số mấy, nếu page là null thì Laravel sẽ xử lý theo data của page query string. Mặc định $page = null.

Trong ví dụ sau, đối số duy nhất được truyền cho phương thức paginate là số items muốn hiển thị "trên mỗi trang". Trong trường hợp này, giả sử muốn hiển thị 15 items trên mỗi trang:

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\DB;
```

```
class UserController extends Controller
{
    /**
     * Show all of the users for the application.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('user.index', [
            'users' => DB::table('users')->paginate(15)
        ]);
    }
}
```

Simple Pagination

Phương thức `paginate` đếm tổng số bản ghi được so khớp bởi truy vấn trước khi truy xuất bản ghi từ cơ sở dữ liệu. Điều này được thực hiện để người phân trang biết tổng cộng có bao nhiêu trang bản ghi. Tuy nhiên, nếu không có kế hoạch hiển thị tổng số trang trong giao diện người dùng của ứng dụng thì truy vấn đếm bản ghi là không cần thiết.

Do đó, nếu chỉ cần hiển thị các liên kết đơn giản "Next" và "Previous" trong giao diện người dùng của ứng dụng, có thể sử dụng phương thức `simplePaginate` để thực hiện một truy vấn hiệu quả:

```
$users = DB::table('users')->simplePaginate(15);
```

1.2.2. Paginating Eloquent Results

Có thể phân trang các truy vấn Eloquent.

Ví dụ này, sẽ phân trang model `App\Models\User` và dự định hiển thị 15 bản ghi trên mỗi trang. Cú pháp gần giống với việc phân trang các kết quả của Query Builder:

```
use App\Models\User;

$users = User::paginate(15);
```

Tất nhiên, có thể gọi phương thức `paginate` sau khi đặt các ràng buộc khác trên truy vấn, chẳng hạn như mệnh đề `where`:

```
$users = User::where('votes', '>', 100)->paginate(15);
```

Cũng có thể sử dụng phương thức `simplePaginate` khi phân trang các Eloquent models:

```
$users = User::where('votes', '>', 100)->simplePaginate(15);
```

Tương tự, có thể sử dụng phương thức `cursorPaginate` để con trỏ phân trang các Eloquent models:

```
$users = User::where('votes', '>', 100)->cursorPaginate(15);
```

Multiple Paginator Instances Per Page

Đôi khi cần hiển thị hai phân trang riêng biệt trên một màn hình duy nhất được ứng dụng hiển thị. Tuy nhiên, nếu cả hai paginator instance sử dụng tham số `page` query string để lưu trữ trang hiện tại, thì hai trình phân trang sẽ xung đột. Để giải quyết xung đột này, có thể chuyển tên của tham số query string muốn sử dụng để lưu trữ trang hiện tại của trình phân trang thông qua đối số thứ ba được cung cấp cho các phương thức `paginate`, `simplePaginate` và `cursorPaginate`:

```
use App\Models\User;

$users = User::where('votes', '>', 100)->paginate(
    $perPage = 15, $columns = ['*'], $pageName = 'users'
);
```

1.2.3. Cursor Pagination

Trong khi `paginate` và `simplePaginate` tạo truy vấn bằng mệnh đề SQL "offset", cursor pagination hoạt động bằng cách xây dựng mệnh đề "where" so sánh giá trị của các cột được sắp xếp có trong truy vấn, cung cấp hiệu suất cơ sở dữ liệu hiệu quả nhất hiện có trong số tất cả các phương thức phân trang của Laravel. Phương thức phân trang này đặc biệt phù hợp với các tập dữ liệu lớn và giao diện người dùng cuộn "infinite".

Không giống như phân trang dựa trên offset, bao gồm một số trang trong chuỗi truy vấn của URL được tạo bởi paginator, phân trang dựa trên cursor đặt một "cursor" string trong chuỗi truy vấn. Cursor là một chuỗi được mã hóa chứa vị trí mà truy vấn được phân trang tiếp theo sẽ bắt đầu phân trang và hướng mà nó sẽ phân trang:

```
http://localhost/users?cursor=eyJpZCI6MTUsIl9wb2ludHNUb05leHRJdGVtcyI6dHJ1ZX0
```

Có thể tạo một cursor dựa trên paginator instance bằng phương thức `cursorPaginate` được cung cấp bởi Query Builder. Phương thức này trả về một instance của `Illuminate\Pagination\CursorPaginator`:

```
$users = DB::table('users')->orderBy('id')->cursorPaginate(15);
```

Khi đã truy xuất một cursor paginator instance, có thể display the pagination results như thường làm khi sử dụng các phương thức `paginate` và `simplePaginate`. Để biết thêm thông tin về các phương thức phiên bản được cung cấp bởi cursor paginator, có thể tham khảo tài liệu về cursor paginator instance method documentation.

Cursor vs. Offset Pagination

Để minh họa sự khác biệt giữa offset pagination và cursor pagination, hãy xem xét một số truy vấn SQL mẫu. Cả hai truy vấn sau đều sẽ hiển thị "second page" của kết quả cho bảng `users` được sắp xếp theo `id`:

```
# Offset Pagination...
select * from users order by id asc limit 15 offset 15;

# Cursor Pagination...
select * from users where id > 15 order by id asc limit 15;
```

Truy vấn cursor pagination cung cấp những ưu điểm sau so với offset pagination:

- Đối với các tập dữ liệu lớn, cursor pagination sẽ mang lại hiệu suất tốt hơn nếu các cột "order by" được lập chỉ mục. Điều này là do mệnh đề "offset" duyệt qua tất cả các dữ liệu đã so khớp trước đó.
- Đối với các tập dữ liệu thường xuyên ghi, offset pagination có thể bỏ qua các bản ghi hoặc hiển thị các bản sao nếu các kết quả gần đây đã được thêm vào hoặc bị xóa khỏi trang mà người dùng hiện đang xem.

Tuy nhiên, cursor pagination có những hạn chế sau:

- Giống như `simplePaginate`, cursor pagination chỉ có thể được sử dụng để hiển thị các liên kết "Next" và "Previous" và không hỗ trợ tạo liên kết với số trang.
- Yêu cầu ordering dựa trên ít nhất một cột unique hoặc kết hợp các cột là unique. Các cột có giá trị null không được hỗ trợ.
- Biểu thức truy vấn trong mệnh đề "order by" chỉ được hỗ trợ nếu chúng được đặt bí danh và cũng

được thêm vào mệnh đề "select".

1.2.4. Manually Creating A Paginator

Có thể tạo một pagination instance theo cách thủ công, bằng cách tạo một instance `Illuminate\Pagination\Paginator`, `Illuminate\Pagination\LengthAwarePaginator` hoặc `Illuminate\Pagination\CursorPaginator`, tùy thuộc vào nhu cầu.

Các lớp `Paginator` và `CursorPaginator` không cần biết tổng số items trong tập kết quả. Do đó, các lớp này không có các phương thức để lấy chỉ mục của trang cuối cùng. `LengthAwarePaginator` chấp nhận các đối số gần giống như `Paginator`; nhưng nó yêu cầu đếm tổng số mục trong tập kết quả.

Nói cách khác, `Paginator` tương ứng với phương thức `simplePaginate` trên Query Builder, `CursorPaginator` tương ứng với phương thức `cursorPaginate` và `LengthAwarePaginator` tương ứng với phương thức `paginate`.

1.2.5. Customizing Pagination URLs

Theo mặc định, các liên kết được tạo bởi paginator sẽ khớp với request's URI hiện tại. Tuy nhiên, phương thức `withPath` của paginator cho phép tùy chỉnh URI được sử dụng bởi paginator khi tạo liên kết.

Ví dụ, nếu muốn paginator tạo các liên kết như `http://example.com/admin/users?page=N`, nên truyền `/admin/users` đến phương thức `withPath`:

```
use App\Models\User;


Route::get('/users', function () {
    $users = User::paginate(15);

    $users->withPath('/admin/users');

    //
});
```

Appending Query String Values

Có thể append vào query string của các pagination links bằng cách sử dụng phương thức `appends`.

Ví dụ, để append `sort=votes` vào mỗi pagination link, nên thực hiện lệnh gọi `appends` như sau:

```
use App\Models\User;
```

```
Route::get('/users', function () {
    $users = User::paginate(15);

    $users->appends(['sort' => 'votes']);

    //
});
```

Có thể sử dụng phương thức `withQueryString` nếu muốn append tất cả các giá trị chuỗi truy vấn của request hiện tại vào các pagination links:

```
$users = User::paginate(15)->withQueryString();
```

Appending Hash Fragments

Nếu cần append "hash fragment" vào các URL được tạo bởi paginator, có thể sử dụng phương thức `fragment`.

Ví dụ, để append `#users` vào cuối mỗi pagination link, nên gọi phương thức `fragment` như sau:

```
$users = User::paginate(15)->fragment('users');
```

1.3. Displaying Pagination Results

Khi gọi phương thức `paginate`, sẽ nhận được một instance của `Illuminate\Pagination\LengthAwarePaginator`, trong khi gọi phương thức `simplePaginate` trả về một instance của `Illuminate\Pagination\Paginator`. Và cuối cùng, việc gọi phương thức `cursorPaginate` trả về một instance của `Illuminate\Pagination\CursorPaginator`.

Các đối tượng này cung cấp một số phương thức mô tả tập hợp kết quả. Ngoài các phương thức trợ giúp này, các instances của trình paginator là các iterators và có thể được lặp lại dưới dạng một mảng. Vì vậy, khi đã truy xuất kết quả, có thể hiển thị kết quả và hiển thị các liên kết trang bằng Blade:

```
<div class="container">
    @foreach ($users as $user)
        {{ $user->name }}
    @endforeach
</div>
```

```
{{ $users->links() }}
```

Phương thức `links` sẽ render các liên kết đến phần còn lại của các trang trong tập kết quả. HTML được tạo bởi phương thức `links` tương thích với khung Tailwind CSS framework.

1.3.1. Adjusting The Pagination Link Window

Khi paginator hiển thị các pagination links, số trang hiện tại được hiển thị cũng như các liên kết cho ba trang trước và sau trang hiện tại. Sử dụng phương thức `onEachSide`, có thể kiểm soát số lượng liên kết bổ sung được hiển thị ở mỗi bên của trang hiện tại trong cửa sổ trượt ở giữa của các liên kết được tạo bởi paginator:

```
{{ $users->onEachSide(5)->links() }}
```

1.3.2. Converting Results To JSON

Các lớp paginator của Laravel thực thi Interface contract `Illuminate\Contracts\Support\Jsonable` và expose phương thức `toJson`, vì vậy rất dễ dàng chuyển đổi kết quả phân trang sang JSON. Có thể chuyển đổi một instance paginator thành JSON bằng cách trả về nó từ một route hoặc controller action:

```
use App\Models\User;

Route::get('/users', function () {
    return User::paginate();
});
```

JSON từ paginator sẽ bao gồm các thông tin như `total`, `current_page`, `last_page`,... Các bản ghi kết quả có sẵn thông qua `data` key trong mảng JSON.

Ví dụ về JSON được tạo bằng cách trả về một phiên bản phân trang từ một route:

```
{
   "total": 50,
   "per_page": 15,
   "current_page": 1,
   "last_page": 4,
   "first_page_url": "http://laravel.app?page=1",
   "last_page_url": "http://laravel.app?page=4",
   "next_page_url": "http://laravel.app?page=2",
```

```
    "prev_page_url": null,
    "path": "http://laravel.app",
    "from": 1,
    "to": 15,
    "data":[
        {
            // Record...
        },
        {
            // Record...
        }
    ]
}
```

1.4. Customizing The Pagination View

Theo mặc định, các v views rendered để hiển thị các pagination links tương thích với Tailwind CSS. Tuy nhiên, nếu không sử dụng Tailwind, có thể tự định nghĩa views để hiển thị các liên kết này. Khi gọi phương thức `links` trên một paginator instance, có thể truyền view name làm đối số đầu tiên cho phương thức:

```
{{ $paginator->links('view.name') }}

// Passing additional data to the view...
{{ $paginator->links('view.name', ['foo' => 'bar']) }}
```

Tuy nhiên, cách dễ nhất để tùy chỉnh các pagination views là xuất chúng sang thư mục `resources/views/vendor` bằng lệnh `vendor:publish`:

```
php artisan vendor:publish --tag=laravel-pagination
```

Lệnh này sẽ đặt các views trong thư mục `resources/views/vendor/pagination` của ứng dụng. File `tailwind.blade.php` trong thư mục này tương ứng với pagination view mặc định. Có thể chỉnh sửa file này để sửa đổi pagination HTML.

Nếu muốn chỉ định một file khác làm pagination view mặc định, có thể gọi các phương thức `defaultView` và `defaultSimpleView` của paginator trong phương thức `boot` của lớp `App\Providers\AppServiceProvider`:

```php
<?php

namespace App\Providers;

use Illuminate\Pagination\Paginator;
use Illuminate\Support\Facades\Blade;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Paginator::defaultView('view-name');

        Paginator::defaultSimpleView('view-name');
    }
}
```

1.4.1. Using Bootstrap

Laravel bao gồm các pagination views được xây dựng bằng Bootstrap CSS. Để sử dụng các views này thay vì Tailwind views mặc định, có thể gọi phương thức `useBootstrap` của paginator trong phương thức `boot` của `App\Providers\AppServiceProvider`:

```php
use Illuminate\Pagination\Paginator;

/**
 * Bootstrap any application services.
 *
```

```
     * @return void
     */
    public function boot()
    {
        Paginator::useBootstrap();
    }
```

1.5. Paginator / LengthAwarePaginator Instance Methods

Mỗi paginator instance cung cấp thông tin phân trang bổ sung thông qua các phương thức sau:

Method	Description
`$paginator->count()`	Lấy số lượng các items cho trang hiện tại.
`$paginator->currentPage()`	Lấy số trang hiện tại.
`$paginator->firstItem()`	Lấy result number của item đầu tiên trong kết quả.
`$paginator->getOptions()`	Lấy các tùy chọn phân trang.
`$paginator->getUrlRange($start, $end)`	Tạo một range của pagination URLs.
`$paginator->hasPages()`	Xác định xem có đủ items để chia thành nhiều trang hay không.
`$paginator->hasMorePages()`	Xác định xem có nhiều items hơn trong data store hay không.
`$paginator->items()`	Lấy các items cho trang hiện tại.
`$paginator->lastItem()`	Lấy result number của item cuối cùng trong kết quả.
`$paginator->lastPage()`	Lấy số trang của trang cuối cùng có sẵn. (Không khả dụng khi sử dụng `simplePaginate`).
`$paginator->nextPageUrl()`	Lấy URL cho trang tiếp theo.
`$paginator->onFirstPage()`	Xác định paginator có ở trang đầu tiên hay không.
`$paginator->perPage()`	Số lượng các items được hiển thị trên mỗi trang.
`$paginator->previousPageUrl()`	Lấy URL cho trang trước.
`$paginator->total()`	Xác định tổng số matching items trong data store. (Không khả dụng khi sử dụng `simplePaginate`).
`$paginator->url($page)`	Lấy URL cho một trang nhất định.

Method	Description
`$paginator->getPageName()`	Lấy biến query string được sử dụng để lưu trữ trang.
`$paginator->setPageName($name)`	Đặt biến query string được sử dụng để lưu trữ trang.

1.6. Cursor Paginator Instance Methods

Mỗi cursor paginator instance cung cấp thông tin phân trang bổ sung thông qua các phương thức sau:

Method	Description
`$paginator->count()`	Lấy số lượng các items cho trang hiện tại.
`$paginator->cursor()`	Lấy cursor instance hiện tại.
`$paginator->getOptions()`	Lấy các tùy chọn phân trang.
`$paginator->hasPages()`	Xác định xem có đủ items để chia thành nhiều trang hay không.
`$paginator->hasMorePages()`	Xác định xem có nhiều items hơn trong data store hay không.
`$paginator->getCursorName()`	Lấy biến query string dùng để lưu cursor.
`$paginator->items()`	Lấy các items cho trang hiện tại.
`$paginator->nextCursor()`	Lấy cursor instance cho tập hợp các items tiếp theo.
`$paginator->nextPageUrl()`	Lấy URL cho trang tiếp theo.
`$paginator->onFirstPage()`	Xác định paginator có ở trang đầu tiên hay không.
`$paginator->perPage()`	Số lượng các items được hiển thị trên mỗi trang.
`$paginator->previousCursor()`	Get the cursor instance for the previous set of items. Lấy cursor instance cho tập hợp các items trước đó.
`$paginator->previousPageUrl()`	Lấy URL cho trang trước.
`$paginator->setCursorName()`	Đặt biến query string được sử dụng để lưu trữ cursor.
`$paginator->url($cursor)`	Lấy URL cho cursor instance nhất định.