



[simonhamp](#) / [AppServiceProvider.php](#)

Last active 2 days ago • Report abuse

 Star

<> Code

 Revisions 13 Stars 239 Forks 43

A pageable Collection implementation for Laravel

 [README.md](#)

Paginated Collections in Laravel

Use Case

Laravel provides pagination out of the box for Eloquent Collections, but you can't use that by default on ordinary Collections.

Collections do have the `forPage()` method, but it's more low-level, so it doesn't generate pagination links.

So you have to create a `LengthAwarePaginator` instance. But what if you want the behaviour to be the same as an Eloquent collection? Then use this macro!

The benefit of this is that the syntax and output is almost identical to the Eloquent Collection `paginate()` method and so it can (relatively) easily be swapped out for an Eloquent Collection when testing.

Installation

Feel free to copy the most relevant code into your project. You're free to use and adapt as you need.

Usage

There are 2 approaches below. Which one you use is up to you, but you don't need both. I personally prefer the macro method as I feel it's cleaner and works well with minimal effort, but it's not so good at working with your IDE (code hints etc) and can feel a little detached in some cases.

The macro way

If you prefer, add the `Collection` macro to a Service Provider. That way you can call `paginate()` on any collection:

```
collect([ ... ])->paginate( 20 );
```

See `AppServiceProvider.php` for a sample implementation.

The subclass way

Where you want a "pageable" collection that is distinct from the standard `Illuminate\Support\Collection`, implement a copy of `Collection.php` in your application and simply replace your `use Illuminate\Support\Collection` statements at the top of your dependent files with `use App\Support\Collection`:

```
// use Illuminate\Support\Collection
use App\Support\Collection;

$items = [];
$collection = (new Collection($items))->paginate(20);
```

Note that this method doesn't work with the `collect()` helper function.

AppServiceProvider.php

```
1  <?php
2
3  namespace App\Providers;
4
5  use Illuminate\Support\Collection;
6  use Illuminate\Pagination\LengthAwarePaginator;
7
8  class AppServiceProvider extends ServiceProvider
9  {
10     public function boot()
11     {
12         /**
13          * Paginate a standard Laravel Collection.
14          *
15          * @param int $perPage
16          * @param int $total
17          * @param int $page
18          * @param string $pageName
19
20          * @return array
21          */
```

```

21     Collection::macro('paginate', function($perPage, $total = null, $page = null, $
22         $page = $page ?: LengthAwarePaginator::resolveCurrentPage($pageName);
23
24         return new LengthAwarePaginator(
25             $this->forPage($page, $perPage),
26             $total ?: $this->count(),
27             $perPage,
28             $page,
29             [
30                 'path' => LengthAwarePaginator::resolveCurrentPath(),
31                 'pageName' => $pageName,
32             ]
33         );
34     });
35 }
36 }

```

Collection.php

```

1  <?php
2
3  namespace App\Support;
4
5  use Illuminate\Pagination\LengthAwarePaginator;
6  use Illuminate\Support\Collection as BaseCollection;
7
8  class Collection extends BaseCollection
9  {
10     public function paginate($perPage, $total = null, $page = null, $pageName = 'page')
11     {
12         $page = $page ?: LengthAwarePaginator::resolveCurrentPage($pageName);
13
14         return new LengthAwarePaginator(
15             $this->forPage($page, $perPage),
16             $total ?: $this->count(),
17             $perPage,
18             $page,
19             [
20                 'path' => LengthAwarePaginator::resolveCurrentPath(),
21                 'pageName' => $pageName,
22             ]
23         );
24     }
25 }

```



defaye commented on May 31, 2017

Fantastic solution simon, thanks.