Laravel Breadcrumbs Documentation

Release 3.0.0

Dave James Miller

17 August 2015

Contents

A simple Laravel-style way to create breadcrumbs in Laravel 5[1].

[1] http://laravel.com/

CHAPTER 1

Table of Contents

1.1 Getting Started

1.1.1 1. Install Laravel Breadcrumbs

Note: Laravel 5.0 or above is required – use the 2.x version[1] for Laravel 4.

Install with Composer

Run this at the command line:

```
$ composer require davejamesmiller/laravel-breadcrumbs
```

This will both update `composer.json` and install the package into the `vendor/` directory.

Add to `config/app.php`

Add the service provider to `providers`:

```
'providers' => [
    // ...
    'DaveJamesMiller\Breadcrumbs\ServiceProvider',
],
```

And add the facade to `aliases`:

```
'aliases' => [
    // ...
    'Breadcrumbs' => 'DaveJamesMiller\Breadcrumbs\Facade',
],
```

1.1.2 2. Define your breadcrumbs

Create a file called `app/Http/breadcrumbs.php` that looks like this:

[1] https://github.com/davejamesmiller/laravel-breadcrumbs/tree/2.x

```php
<?php

// Home
Breadcrumbs::register('home', function($breadcrumbs)
{
    $breadcrumbs->push('Home', route('home'));
});

// Home > About
Breadcrumbs::register('about', function($breadcrumbs)
{
    $breadcrumbs->parent('home');
    $breadcrumbs->push('About', route('about'));
});

// Home > Blog
Breadcrumbs::register('blog', function($breadcrumbs)
{
    $breadcrumbs->parent('home');
    $breadcrumbs->push('Blog', route('blog'));
});

// Home > Blog > [Category]
Breadcrumbs::register('category', function($breadcrumbs, $category)
{
    $breadcrumbs->parent('blog');
    $breadcrumbs->push($category->title, route('category', $category->id));
});

// Home > Blog > [Category] > [Page]
Breadcrumbs::register('page', function($breadcrumbs, $page)
{
    $breadcrumbs->parent('category', $page->category);
    $breadcrumbs->push($page->title, route('page', $page->id));
});
```

See the Defining Breadcrumbs (page 5) section for more details.

1.1.3 3. Choose a template

By default a Bootstrap[2]-compatible ordered list will be rendered, so if you're using Bootstrap 3 you can skip this step.

First initialise the config file by running this command:

```
$ php artisan vendor:publish
```

Then open `config/breadcrumbs.php` and edit this line:

```
'view' => 'breadcrumbs::bootstrap3',
```

The possible values are:

- Bootstrap 3[3]: `breadcrumbs::bootstrap3`
- Bootstrap 2[4]: `breadcrumbs::bootstrap2`

[2] http://getbootstrap.com/components/#breadcrumbs
[3] http://getbootstrap.com/components/#breadcrumbs
[4] http://getbootstrap.com/2.3.2/components.html#breadcrumbs

- The path to a custom view: e.g. `_partials/breadcrumbs`

See the Custom Templates (page 7) section for more details.

1.1.4 4. Output the breadcrumbs

Finally, call `Breadcrumbs::render()` in the view template for each page, passing it the name of the breadcrumb to use and any additional parameters – for example:

```
{!! Breadcrumbs::render('home') !!}

{!! Breadcrumbs::render('category', $category) !!}
```

See the Outputting Breadcrumbs (page 7) section for other output options, and see Route-Bound Breadcrumbs (page 8) for a way to link breadcrumb names to route names automatically.

1.2 Defining Breadcrumbs

Breadcrumbs will usually correspond to actions or types of page. For each breadcrumb you specify a name, the breadcrumb title and the URL to link it to. Since these are likely to change dynamically, you do this in a closure, and you pass any variables you need into the closure.

The following examples should make it clear:

1.2.1 Static pages

The most simple breadcrumb is probably going to be your homepage, which will look something like this:

```
Breadcrumbs::register('home', function($breadcrumbs) {
    $breadcrumbs->push('Home', route('home'));
});
```

As you can see, you simply call `$breadcrumbs->push($title, $url)` inside the closure.

For generating the URL, you can use any of the standard Laravel URL-generation methods, including:

- `url('path/to/route')` (`URL::to()`)
- `secure_url('path/to/route')`
- `route('routename')` or `route('routename', 'param')` or `route('routename', ['param1', 'param2'])` (`URL::route()`)
- `action('controller@action')` (`URL::action()`)
- Or just pass a string URL (`'http://www.example.com/'`)

This example would be rendered like this:

```
Home
```

1.2.2 Parent links

This is another static page, but this has a parent link before it:

```
Breadcrumbs::register('blog', function($breadcrumbs) {
    $breadcrumbs->parent('home');
    $breadcrumbs->push('Blog', route('blog'));
});
```

It would be rendered like this:

```
Home > Blog
```

1.2.3 Dynamic titles and links

This is a dynamically generated page pulled from the database:

```
Breadcrumbs::register('page', function($breadcrumbs, $page) {
    $breadcrumbs->parent('blog');
    $breadcrumbs->push($page->title, route('page', $page->id));
});
```

The `$page` variable would simply be passed in from the view:

```
{!! Breadcrumbs::render('page', $page) !!}
```

It would be rendered like this:

```
Home > Blog > Page Title
```

Tip: You can pass multiple parameters if necessary.

1.2.4 Nested categories

Finally if you have nested categories or other special requirements, you can call `$breadcrumbs->push()` multiple times:

```
Breadcrumbs::register('category', function($breadcrumbs, $category) {
    $breadcrumbs->parent('blog');

    foreach ($category->ancestors as $ancestor) {
        $breadcrumbs->push($ancestor->title, route('category', $ancestor->id));
    }

    $breadcrumbs->push($category->title, route('category', $category->id));
});
```

Alternatively you could make a recursive function such as this:

```
Breadcrumbs::register('category', function($breadcrumbs, $category) {
    if ($category->parent)
        $breadcrumbs->parent('category', $category->parent);
    else
        $breadcrumbs->parent('blog');

    $breadcrumbs->push($category->title, route('category', $category->slug));
});
```

Both would be rendered like this:

```
Home > Blog > Grandparent Category > Parent Category > Category Title
```

1.3 Custom Templates

1.3.1 Create a view

To customise the HTML, create your own view file (e.g. `resources/views/_partials/breadcrumbs.blade.php`) like this:

```
@if ($breadcrumbs)
    <ul class="breadcrumb">
        @foreach ($breadcrumbs as $breadcrumb)
            @if (!$breadcrumb->last)
                <li><a href="{{{ $breadcrumb->url }}}">{{{ $breadcrumb->title }}}</a></li>
            @else
                <li class="active">{{{ $breadcrumb->title }}}</li>
            @endif
        @endforeach
    </ul>
@endif
```

(See the views/ directory[5] for the built-in templates.)

View data

The view will receive an array called `$breadcrumbs`.

Each breadcrumb is an object with the following keys:

- `title` - The title you set above
- `url` - The URL you set above
- `first` - `true` for the first breadcrumb (top level), `false` otherwise
- `last` - `true` for the last breadcrumb (current page), `false` otherwise
- Additional keys for each item in `$data` (see *Custom data* (page 11))

1.3.2 Update the config

Then update your config file (`config/breadcrumbs.php`) with the custom view name, e.g.:

```
'view' => '_partials/breadcrumbs',
```

1.4 Outputting Breadcrumbs

Call `Breadcrumbs::render()` in the view template for each page, passing it the name of the breadcrumb to use and any additional parameters.

[5] https://github.com/davejamesmiller/laravel-breadcrumbs/tree/master/views

1.4.1 With Blade

In the page (e.g. `resources/views/home.blade.php`):

```
{!! Breadcrumbs::render('home') !!}
```

Or with a parameter:

```
{!! Breadcrumbs::render('category', $category) !!}
```

1.4.2 With Blade layouts and @section

In the page (e.g. `resources/views/home.blade.php`):

```
@extends('layout.name')

@section('breadcrumbs', Breadcrumbs::render('home'))
```

In the layout (e.g. `resources/views/app.blade.php`):

```
@yield('breadcrumbs')
```

1.4.3 Pure PHP (without Blade)

In the page (e.g. `resources/views/home.php`):

```
<?= Breadcrumbs::render('home') ?>
```

Or use the long-hand syntax if you prefer:

```
<?php echo Breadcrumbs::render('home') ?>
```

1.5 Route-Bound Breadcrumbs

In normal usage you must call `Breadcrumbs::render($name, $params...)` to render the breadcrumbs on every page. If you prefer, you can name your breadcrumbs the same as your routes and avoid this duplication.

1.5.1 Setup

Name your routes

Make sure each of your routes has a name (`'as'` parameter). For example (`app/Http/routes.php`):

```
// Home
Route::get('/', ['as' => 'home', 'uses' => 'HomeController@index']);

// Home > [Page]
Route::get('/page/{id}', ['as' => 'page', 'uses' => 'PageController@show']);
```

For more details see Named Routes[6] in the Laravel documentation.

[6] http://laravel.com/docs/routing#named-routes

Name your breadcrumbs to match

For each route, create a breadcrumb with the same name. For example (`app/Http/routes.php`):

```
// Home
Breadcrumbs::register('home', function($breadcrumbs) {
    $breadcrumbs->push('Home', route('home'));
});

// Home > [Page]
Breadcrumbs::register('page', function($breadcrumbs, $id)
{
    $page = Page::findOrFail($id);
    $breadcrumbs->parent('home');
    $breadcrumbs->push($page->title, route('page', $page->id));
});
```

Output breadcrumbs in your layout

Call `Breadcrumbs::render()` with no parameters in your layout file (e.g. `resources/views/app.blade.php`):

```
{!! Breadcrumbs::render() !!}
```

This will automatically output breadcrumbs corresponding to the current route.

It will throw an exception if the breadcrumb doesn't exist, to remind you to create one. To prevent this behaviour, change it to:

```
{!! Breadcrumbs::renderIfExists() !!}
```

1.5.2 Route model binding

Laravel Breadcrumbs uses the same model binding as the controller. For example:

```
// app/Http/routes.php
Route::model('page', 'Page');
Route::get('/page/{page}', ['uses' => 'PageController@show', 'as' => 'page']);
```

```
// app/Http/Controllers/PageController.php
class PageController extends Controller {
    public function show($page)
    {
        return view('page/show', ['page' => $page]);
    }
}
```

```
// app/Http/breadcrumbs.php
Breadcrumbs::register('page', function($breadcrumbs, $page)
{
    $breadcrumbs->parent('home');
    $breadcrumbs->push($page->title, route('page', $page->id));
});
```

This makes your code less verbose and more efficient by only loading the page from the database once.

For more details see Route Model Binding[7] in the Laravel documentation.

1.5.3 Resourceful controllers

Laravel automatically creates route names for resourceful controllers, e.g. `photo.index`, which you can use when defining your breadcrumbs. For example:

```
// app/Http/routes.php
Route::resource('photo', 'PhotoController');
```

```
$ php artisan route:list
+--------+----------+--------------------+---------------+--------------------------+------------+
| Domain | Method   | URI                | Name          | Action                   | Middleware |
+--------+----------+--------------------+---------------+--------------------------+------------+
|        | GET|HEAD | photo              | photo.index   | PhotoController@index    |            |
|        | GET|HEAD | photo/create       | photo.create  | PhotoController@create   |            |
|        | POST     | photo              | photo.store   | PhotoController@store    |            |
|        | GET|HEAD | photo/{photo}      | photo.show    | PhotoController@show     |            |
|        | GET|HEAD | photo/{photo}/edit | photo.edit    | PhotoController@edit     |            |
|        | PUT      | photo/{photo}      | photo.update  | PhotoController@update   |            |
|        | PATCH    | photo/{photo}      |               | PhotoController@update   |            |
|        | DELETE   | photo/{photo}      | photo.destroy | PhotoController@destroy  |            |
+--------+----------+--------------------+---------------+--------------------------+------------+
```

```
// app/Http/breadcrumbs.php

// Photos
Breadcrumbs::register('photo.index', function($breadcrumbs)
{
    $breadcrumbs->parent('home');
    $breadcrumbs->push('Photos', route('photo.index'));
});

// Photos > Upload Photo
Breadcrumbs::register('photo.create', function($breadcrumbs)
{
    $breadcrumbs->parent('photo.index');
    $breadcrumbs->push('Upload Photo', route('photo.create'));
});

// Photos > [Photo Name]
Breadcrumbs::register('photo.show', function($breadcrumbs, $photo)
{
    $breadcrumbs->parent('photo.index');
    $breadcrumbs->push($photo->title, route('photo.show', $photo->id));
});

// Photos > [Photo Name] > Edit Photo
Breadcrumbs::register('photo.edit', function($breadcrumbs, $photo)
{
    $breadcrumbs->parent('photo.show', $photo);
    $breadcrumbs->push('Edit Photo', route('photo.edit', $photo->id));
});
```

For more details see RESTful Resource Controllers[8] in the Laravel documentation.

[7] http://laravel.com/docs/routing#route-model-binding
[8] http://laravel.com/docs/controllers#restful-resource-controllers

1.5.4 Implicit controllers

To use implicit controllers, you must specify names for each route. For example:

```
// app/Http/routes.php
Route::controller('auth', 'Auth\AuthController', [
    'getRegister' => 'auth.register',
    'getLogin'    => 'auth.login',
]);
```

(You don't need to provide route names for actions that redirect and never display a view - e.g. most POST views.)

For more details see Implicit Controllers[9] in the Laravel documentation.

1.6 Advanced Usage

1.6.1 Breadcrumbs with no URL

The second parameter to `push()` is optional, so if you want a breadcrumb with no URL you can do so:

```
$breadcrumbs->push('Sample');
```

The `$breadcrumb->url` value will be `null`.

The default Twitter Bootstrap templates provided render this with a CSS class of "active", the same as the last breadcrumb, because otherwise they default to black text not grey which doesn't look right.

Note: Support for this was added to the Twitter Bootstrap templates in 2.1.0. Before this you would need to create a custom template.

1.6.2 Custom data

Added in 2.3.0.

The `push()` method accepts an optional third parameter, `$data` - an array of arbitrary data to be passed to the breadcrumb, which you can use in your custom template. For example, if you wanted each breadcrumb to have an icon, you could do:

```
$breadcrumbs->push('Home', '/', ['icon' => 'home.png']);
```

The `$data` array's entries will be merged into the breadcrumb as properties, so you would access the icon as `$breadcrumb->icon` in your template, like this:

```
<li><a href="{{{ $breadcrumb->url }}}">
    <img src="/images/icons/{{{ $breadcrumb->icon }}}">
    {{{ $breadcrumb->title }}}
</a></li>
```

Do not use the following keys in your data array, as they will be overwritten: `title`, `url`, `first`, `last`.

1.6.3 Defining breadcrumbs in a different file

If you don't want to use `app/Http/breadcrumbs.php`, you can define them in `app/Http/routes.php` or any other file as long as it's loaded by Laravel.

[9] http://laravel.com/docs/controllers#implicit-controllers

1.6.4 Switching views dynamically

You can change the view at runtime by calling:

```
Breadcrumbs::setView('view.name');
```

Or you can call `Breadcrumbs::generate()` and then load the view manually:

```
@include('_partials/breadcrumbs2', ['breadcrumbs' => Breadcrumbs::generate('category', $category)])
```

1.6.5 Overriding the "current" route

If you call `Breadcrumbs::render()` or `Breadcrumbs::generate()` with no parameters, it will use the current route name and parameters by default (as returned by Laravel's `Route::current()` method).

You can override this by calling `Breadcrumbs::setCurrentRoute($name, $param1, $param2...)` or `Breadcrumbs::setCurrentRouteArray($name, $params)`.

1.6.6 Passing an array of parameters

Added in 2.0.0.

If the breadcrumb requires multiple parameters, you would normally pass them like this:

```
Breadcrumbs::render('name', $param1, $param2, $param3);
Breadcrumbs::generate('name', $param1, $param2, $param3);
$breadcrumbs->parent('name', $param1, $param2, $param3);
```

If you want to pass an array of parameters instead you can use these methods:

```
Breadcrumbs::renderArray('name', $params);
Breadcrumbs::generateArray('name', $params);
$breadcrumbs->parentArray('name', $params);
```

1.6.7 Checking if a breadcrumb exists

Added in 2.2.0.

By default an exception will be thrown if the breadcrumb doesn't exist, so you know to add it. If you want suppress this you can call the following methods instead:

- `Breadcrumbs::renderIfExists()` (returns an empty string)
- `Breadcrumbs::renderArrayIfExists()` (returns an empty string)
- `Breadcrumbs::generateIfExists()` (returns an empty array)
- `Breadcrumbs::generateArrayIfExists()` (returns an empty array)

Alternatively you can call `Breadcrumbs::exists('name')`, which returns a boolean.

1.7 API Reference

1.7.1 `Breadcrumbs` Facade

Method	Returns	Added in	Docs
`Breadcrumbs::register($name, $callback)`	*(none)*	1.0.0	Defining (page 5)
`Breadcrumbs::exists()`	boolean	2.2.0	Route-bound (page 8)
`Breadcrumbs::exists($name)`	boolean	2.2.0	*Exists* (page 12)
`Breadcrumbs::generate()`	array	2.2.3	Route-bound (page 8)
`Breadcrumbs::generate($name)`	array	1.0.0	*Switching views* (page 12)
`Breadcrumbs::generate($name, $param1, ...)`	array	1.0.0	*Switching views* (page 12)
`Breadcrumbs::generateArray($name, $params)`	array	2.0.0	*Array params* (page 12)
`Breadcrumbs::generateIfExists()`	array	2.2.0	Route-bound (page 8)
`Breadcrumbs::generateIfExists($name)`	array	2.2.0	*Exists* (page 12)
`Breadcrumbs::generateIfExists($name, $param1, ...)`	array	2.2.0	*Exists* (page 12)
`Breadcrumbs::generateIfExistsArray($name, $params)`	array	3.0.0	*Exists* (page 12)
`Breadcrumbs::render()`	string	2.2.0	Route-bound (page 8)
`Breadcrumbs::render($name)`	string	1.0.0	Output (page 7)
`Breadcrumbs::render($name, $param1, ...)`	string	1.0.0	Output (page 7)
`Breadcrumbs::renderArray($name, $params)`	string	2.0.0	*Array params* (page 12)
`Breadcrumbs::renderIfExists()`	string	2.2.0	Route-bound (page 8)
`Breadcrumbs::renderIfExists($name)`	string	2.2.0	*Exists* (page 12)
`Breadcrumbs::renderIfExists($name, $param1, ...)`	string	2.2.0	*Exists* (page 12)
`Breadcrumbs::renderIfExistsArray($name, $params)`	string	3.0.0	*Exists* (page 12)
`Breadcrumbs::setCurrentRoute($name)`	*(none)*	2.2.0	*Current route* (page 12)
`Breadcrumbs::setCurrentRoute($name, $param1, ...)`	*(none)*	2.2.0	*Current route* (page 12)
`Breadcrumbs::setCurrentRouteArray($name, $params)`	*(none)*	2.2.0	*Current route* (page 12)
`Breadcrumbs::clearCurrentRoute()`	*(none)*	2.2.0	
`Breadcrumbs::setView($view)`	*(none)*	1.0.0	*Switching views* (page 12)

Source[10]

1.7.2 Defining breadcrumbs

```
Breadcrumbs::register('name', function($breadcrumbs, $page) {
    // ...
});
```

[10] https://github.com/davejamesmiller/laravel-breadcrumbs/blob/develop/src/Manager.php

Method	Returns	Added in	Docs
`$breadcrumbs->push($title)`	*(none)*	1.0.0	*No URL* (page 11)
`$breadcrumbs->push($title, $url)`	*(none)*	1.0.0	Defining (page 5)
`$breadcrumbs->push($title, $url, $data)`	*(none)*	2.3.0	*Custom data* (page 11)
`$breadcrumbs->parent($name)`	*(none)*	1.0.0	*Parent links* (page 5)
`$breadcrumbs->parent($name, $param1, ...)`	*(none)*	1.0.0	*Parent links* (page 5)
`$breadcrumbs->parentArray($name, $params)`	*(none)*	2.0.0	*Array parameters* (page 12)

Source[11]

1.7.3 In the view (template)

`$breadcrumbs` (array), contains:

Variable	Type	Added in	Docs
`$breadcrumb->title`	string	1.0.0	*View data* (page 7)
`$breadcrumb->url`	string or null	1.0.0	*View data* (page 7)
`$breadcrumb->first`	boolean	1.0.0	*View data* (page 7)
`$breadcrumb->last`	boolean	1.0.0	*View data* (page 7)
`$breadcrumb->custom_attribute_name`	mixed	2.3.0	*Custom data* (page 11)

1.8 Contributing

If you want to submit a **bug fix**, make your changes in a new branch, based on the `develop` branch, then simply open a pull request[12] on GitHub. (The information below may help you to get started if you've not done this before.)

If you want to submit a **new feature**, it's usually best to open an issue[13] to discuss the idea first – to make sure it will be accepted before spending too much time on it. (Of course you can go ahead and develop it first if you prefer!) Please be sure to update the documentation as well.

1.8.1 Developing inside a real application

The easiest way to develop Laravel Breadcrumbs alongside a real Laravel application is to set it up as normal, but tell Composer to install from source with the `--prefer-source` flag.

If you've already got it installed, delete it from the `vendor/` directory and re-install from source:

```
$ cd /path/to/repo
$ rm -rf vendor/davejamesmiller/laravel-breadcrumbs
$ composer install --prefer-source
$ cd vendor/davejamesmiller/laravel-breadcrumbs
$ git checkout -t origin/develop
$ git checkout -b YOUR_BRANCH
# Make changes and commit them
$ git remote add YOUR_USERNAME git@github.com:YOUR_USERNAME/laravel-breadcrumbs
$ git push -u YOUR_USERNAME YOUR_BRANCH
```

[11] https://github.com/davejamesmiller/laravel-breadcrumbs/blob/develop/src/Generator.php
[12] https://github.com/davejamesmiller/laravel-breadcrumbs/pulls
[13] https://github.com/davejamesmiller/laravel-breadcrumbs/issues

There is also a test app[14] available to simplify testing against multiple versions of Laravel.

Note: The test app is not a replacement for unit tests - we have those too - but it gives a better feel for how the package works in practice.

1.8.2 Using your fork in a project

If you have forked the package (e.g. to fix a bug or add a feature), you may want to use that version in your project until the changes are merged and released. To do that, simply update the `composer.json` in your main project as follows:

```
{
    "repositories": [
        {
            "type": "vcs",
            "url": "https://github.com/YOUR_USERNAME/laravel-breadcrumbs.git"
        }
    ],
    "require": {
        "davejamesmiller/laravel-breadcrumbs": "dev-YOUR_BRANCH"
    }
}
```

Replace `YOUR_USERNAME` with your GitHub username and `YOUR_BRANCH` with the branch name (e.g. `develop`). This tells Composer to use your repository instead of the default one.

1.8.3 Unit tests

To run the unit tests, simply run:

```
$ cd /path/to/laravel-breadcrumbs
$ composer update
$ ./test.sh
```

(Note: The unit tests are not 100% complete yet, and the code will probably need some refactoring to make it easier to test.)

Code coverage

To check code coverage, you will also need Xdebug[15] installed. Run:

```
$ ./test-coverage.sh
```

Then open `test-coverage/index.html` to view the results. (However, be aware of the edge cases[16] in PHPUnit that can make it not-quite-accurate.)

[14] https://github.com/davejamesmiller/laravel-breadcrumbs-test
[15] http://xdebug.org/
[16] https://phpunit.de/manual/current/en/code-coverage-analysis.html#code-coverage-analysis.edge-cases

1.8.4 Documentation

Documentation is in `docs/`. It is written in reStructuredText[17] and converted to HTML and PDF formats by Sphinx[18].

To submit a documentation change, simply edit the appropriate file on GitHub[19]. (There's an "Edit on GitHub" link in the top-right corner of each page.)

> **Warning:** Not all markup is supported by GitHub – e.g. `:ref:` and `:doc:` – so the preview may not be exactly what appears in the online documentation.

For more comprehensive documentation changes you may be better installing Sphinx so you can test the docs locally:

Installing Sphinx

You will need Python[20] and pip[21] to install Sphinx[22], the documentation generator. To install them (on Debian Wheezy or similar), you can run the following:

```
$ sudo apt-get install python python-pip
$ sudo pip install sphinx sphinx-autobuild sphinx_rtd_theme
```

To build the PDF documentation, you will also need LaTeX installed:

```
$ sudo apt-get install texlive texlive-latex-extra
```

Building documentation

To build the HTML docs (`docs-html/index.html`):

```
$ ./build-html-docs.sh
```

This will build the docs and run a HTML server on port 8000 that will automatically rebuild the docs and reload the page whenever you modify a file.

To build the PDF docs (`docs-pdf/laravel-breadcrumbs.pdf`):

```
$ ./build-pdf-docs.sh
```

Sphinx markup reference

I found the following documents useful when writing the documentation:

- reStructuredText quick reference[23]
- Admonitions list[24] (`note::`, `warning::`, etc.)
- Code examples markups[25] (`code-block::`, `highlight::`)
- Other paragraph-level markup[26] (`versionadded::`, `deprecated::`, etc.)

[17] http://docutils.sourceforge.net/rst.html
[18] http://sphinx-doc.org/
[19] https://github.com/alberon/awe/tree/master/docs
[20] https://www.python.org/
[21] https://pypi.python.org/pypi/pip
[22] http://sphinx-doc.org/
[23] http://docutils.sourceforge.net/docs/user/rst/quickref.html
[24] http://docutils.sourceforge.net/docs/ref/rst/directives.html#admonitions
[25] http://sphinx-doc.org/markup/code.html
[26] http://sphinx-doc.org/markup/para.html

- Inline markup[27] (`:ref:`, `:doc:`, etc.)
- Table of contents[28] (`toctree::`)

Heading styles

The following code styles are used for headings:

```
##############################################################################
 Page title (80 hashes)
##############################################################################

==============================================================================
 Section title (80 equals signs)
==============================================================================

------------------------------------
 Heading 2 (40 hypens)
------------------------------------

Heading 3 (full stops)
......................
```

1.9 Changelog

Laravel Breadcrumbs uses Semantic Versioning[29].

1.9.1 v3.0.0 (8 Feb 2015)

- Add Laravel 5 support (#62[30])
- Change view namespace from `laravel-breadcrumbs::` to `breadcrumbs::`
- Change Bootstrap 3 template from `<ul>` to `<ol>` to match the documentation[31]
- Move documentation from GitHub (Markdown) to Read The Docs[32] (reStructuredText/Sphinx[33])
- Greatly improve unit & integration tests (largely thanks to Testbench[34])
- Fix issue that prevented non-deferred service providers referencing Breadcrumbs (#39[35]) by making Breadcrumbs non-deferred also
- Rename `generateArrayIfExists()` to `generateIfExistsArray()`
- Rename `renderArrayIfExists()` to `renderIfExistsArray()`
- Remove `$breadcrumbs->get()` and `$breadcrumbs->set()` methods from Generator class (they were never used nor documented)

[27] http://sphinx-doc.org/markup/inline.html
[28] http://sphinx-doc.org/markup/toctree.html
[29] http://semver.org/
[30] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/62
[31] http://getbootstrap.com/components/#breadcrumbs
[32] https://readthedocs.org/
[33] http://sphinx-doc.org/
[34] https://github.com/orchestral/testbench
[35] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/39

- Remove `Breadcrumbs::getView()`
- Switch from PSR-0 to PSR-4 file naming

Upgrading from 2.x to 3.x

- Upgrade to Laravel 5[36]
- Move `app/breadcrumbs.php` to `app/Http/breadcrumbs.php`
- Move `app/config/packages/davejamesmiller/laravel-breadcrumbs/config.php` to `config/breadcrumbs.php` (if used)

The following changes are optional because there are shims in place:

- In the config file, replace `laravel-breadcrumbs::` with `breadcrumbs::`
- Replace any calls to `Breadcrumbs::generateArrayIfExists()` with `Breadcrumbs::generateIfExistsArray()`
- Replace any calls to `Breadcrumbs::renderArrayIfExists()` with `Breadcrumbs::renderIfExistsArray()`

Note: Laravel 4 and PHP 5.3 are no longer supported – please continue to use the 2.x branch[37] if you use them.

1.9.2 v2.3.1 (8 Feb 2015)

- Fix issue that prevented non-deferred service providers referencing Breadcrumbs (#39[38]) by making Breadcrumbs non-deferred also (backported from 3.0.0)

1.9.3 v2.3.0 (26 Oct 2014)

- Add `$data` parameter to `$breadcrumb->push()` to allow for arbitrary data (#34[39], #35[40], #55[41], #56[42])

1.9.4 v2.2.3 (10 Sep 2014)

- Fix `Breadcrumbs::generate()` with no parameters so it uses the current route, like `Breadcrumbs::render()` does (#46[43])

1.9.5 v2.2.2 (3 Aug 2014)

- Support for Laravel's `App::missing()` method when using automatic route detection (#40[44], #41[45])

[36] http://laravel.com/docs/5.0/upgrade#upgrade-5.0
[37] https://github.com/davejamesmiller/laravel-breadcrumbs/tree/2.x
[38] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/39
[39] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/34
[40] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/35
[41] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/55
[42] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/56
[43] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/46
[44] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/40
[45] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/41

1.9.6 v2.2.1 (19 May 2014)

- Laravel 4.2 support (#21[46], #28[47])

1.9.7 v2.2.0 (26 Jan 2014)

- Add `Breadcrumbs::exists()`, `renderIfExists()`, `renderArrayIfExists()` (#22[48])
- Use the current route name & parameters by default so you don't have to specify them in the view (as long as you use consistent names) (#16[49], #24[50])

1.9.8 v2.1.0 (16 Oct 2013)

- Add support for non-linked breadcrumbs to the Twitter Bootstrap templates (#20[51])

1.9.9 v2.0.0 (28 Sep 2013)

- Add Twitter Bootstrap v3 template (#7[52])
- Twitter Bootstrap v3 is now the default template
- Support for passing arrays into `render()`, `generate()` and `parent()` (**not backwards-compatible**) (#8[53])
  - Split `Breadcrumbs::render()` into `render($name, $arg1, $arg2)` and `renderArray($name, $params)`
  - Split `Breadcrumbs::generate()` into `generate($name, $arg1, $arg2)` and `generateArray($name, $params)`
  - Split `$breadcrumbs->parent()` into `parent($name, $arg1, $arg2)` and `parentArray($name, $params)`
- Set view name in config file instead of in `breadcrumbs.php` (#10[54], #11[55])
- Simplify class names (#15[56])
- Add unit tests

Upgrading from 1.x to 2.x

- In `app/config/app.php` change `DaveJamesMiller\Breadcrumbs\BreadcrumbsServiceProvider` to `DaveJamesMiller\Breadcrumbs\ServiceProvider`
- In `app/config/app.php` change `DaveJamesMiller\Breadcrumbs\Facades\Breadcrumbs` to `DaveJamesMiller\Breadcrumbs\Facade`

[46] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/21
[47] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/28
[48] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/22
[49] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/16
[50] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/24
[51] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/20
[52] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/7
[53] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/8
[54] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/10
[55] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/11
[56] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/15

- The default template was changed from Bootstrap 2 to Bootstrap 3. See *Choose a template* (page 4) if you need to switch it back.

The following internal changes will not affect most people but if you have any problems please be aware of the following:

- The view namespace was changed from `breadcrumbs` to `laravel-breadcrumbs` to match the Composer project name.
- The Bootstrap 2 template name was changed from `breadcrumbs::bootstrap` to `laravel-breadcrumbs::bootstrap2`.
- If you pass arrays into any of the methods, please read the following section:

Passing arrays into `render()`, `generate()` and `parent()`

In **version 1.x** you could pass an array into each of these methods and it was split up into several parameters. For example:

```
// If this breadcrumb is defined:
Breadcrumbs::register('page', function($breadcrumbs, $param1, $param2)
{
    $breadcrumbs->push($param1, $param2);
});

// Then this:
Breadcrumbs::render('page', ['param1', 'param2']);

// Was equivalent to this:
Breadcrumbs::render('page', 'param1', 'param2');

// But to pass an array as the first parameter you would have to do this instead:
Breadcrumbs::render('page', [['param1A', 'param1B']]);
```

This means you couldn't pass an array as the first parameter unless you wrapped all parameters in another array (issue #8[57]).

In **version 2.x** this has been split into two methods:

```
// Now this:
Breadcrumbs::renderArray('page', ['param1', 'param2']);

// Is equivalent to this:
Breadcrumbs::render('page', 'param1', 'param2');

// And this only passes a single parameter (an array) to the callback:
Breadcrumbs::render('page', ['param1A', 'param1B']);
```

Similarly `Breadcrumbs::generateArray()` and `$breadcrumbs->parentArray()` methods are available, which take a single array argument.

1.9.10 v1.0.1 (13 Jul 2013)

- Fix for PHP 5.3 compatibility (#3[58])

[57] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/8
[58] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/3

1.9.11 v1.0.0 (25 May 2013)

- Initial release

1.10 Thanks to

This package is largely based on the Gretel[59] plugin for Ruby on Rails, which I used for a while before discovering Laravel.

And of course it would be nothing without Laravel[60] itself!

1.10.1 Contributors

- Christian Thomas (christian-thomas[61]) - #62[62]
- Miloš Levačić (levacic[63]) - #56[64]
- Ricky Wiens (rickywiens[65]) - #41[66]
- Boris Glumpler (shabushabu[67]) - #28[68]
- Andrej Badin (Andreyco[69]) - #24[70]
- Stef Horner (tedslittlerobot[71]) - #11[72]

1.11 License

The MIT License (MIT)[73]

[59] https://github.com/lassebunk/gretel
[60] http://laravel.com/
[61] https://github.com/christian-thomas
[62] https://github.com/davejamesmiller/laravel-breadcrumbs/issues/62#issuecomment-71724019
[63] https://github.com/levacic
[64] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/56
[65] https://github.com/rickywiens
[66] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/41
[67] https://github.com/shabushabu
[68] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/28
[69] https://github.com/Andreyco
[70] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/24
[71] https://github.com/tedslittlerobot
[72] https://github.com/davejamesmiller/laravel-breadcrumbs/pull/11
[73] http://choosealicense.com/licenses/mit/