# [Add Your Custom Helpers in Laravel 5](http://slick.pl/kb/laravel/add-your-custom-helpers-in-laravel-5/)

Recently I reached to the point in my Laravel 5 application where I’ve started repeating some small parts of my code in many places. I knew that in the future it can easily get out of control when there will be a need to change something. Obviouslly solution like “search and replace” is something we even should not consider for a moment. Instead of repeating myself I created helper functions.

Helper functions are nothing else as your custom standalone PHP functions that you just include and call whereever it’s a must. Then, if you have to change anything, you do it just in one place – the function’s definition. Below I will show you, how I implemented custom helper functions in Laravel 5. If you find my way as inappropriate or completely wrong, please let me know. We all learn everyday and I know that in Laravel you can do one things in multiple ways.

Before we jump to the code, let’s think for a while about arranging our helper function in more organised way. What I did?

1. Inside **app** directory I created another directory called **Helpers**.
2. Inside **Helpers** I created individual files for functions that are designed to do some specific things (of course you’re free to change this – maybe it’s better for you just to create one single file called **Functions.php**):
   1. **Text.php** – for manipulating stings,
   2. **Avatar.php** – for displaying user’s avatar,
   3. **General.php** – for common and “general” things like returning some specific arrays, converting bytes to more “readable format” using kB, MB etc.
3. Next I wrote a Service Provider that is responsible for including my functions.
4. Last step is to edit **app.php** inside **config** directory and add specific code into “providers” and “aliases” arrays.

Let’s start. Inside

|  |  |
| --- | --- |
| 1 | app |

directory create another one called

|  |  |
| --- | --- |
| 1 | Helpers |

Next, inside create new file called

|  |  |
| --- | --- |
| 1 | Text.php |

Open this file and paste below code

|  |  |  |
| --- | --- | --- |
| 01 | <?php namespace Darwin\Helpers; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | class Text { | |
| 04 |  |

|  |  |
| --- | --- |
| 05 | /\*\* |
| 06 | \* Returns an excerpt from a given string (between 0 and passed limit variable). | |

|  |  |
| --- | --- |
| 07 | \* |
| 08 | \* @param $string | |

|  |  |
| --- | --- |
| 09 | \* @param int $limit |
| 10 | \* @param string $suffix | |

|  |  |  |
| --- | --- | --- |
| 11 | \* @return string | |
| 12 | \*/ |

|  |  |  |
| --- | --- | --- |
| 13 | public static function shorten($string, $limit = 100, $suffix = '…') | |
| 14 | { |

|  |  |  |
| --- | --- | --- |
| 15 | if (strlen($string) < $limit) { | |
| 16 | return $string; |

|  |  |  |
| --- | --- | --- |
| 17 | } | |
| 18 |  |

|  |  |  |
| --- | --- | --- |
| 19 | return substr($string, 0, $limit) . $suffix; | |
| 20 | } |

|  |  |
| --- | --- |
| 21 | } |

Change the namespace to the one you use. As you see, we’ve defined one public static function called **shorten**. Keep in mind that this is just an example. We’ll ignore **Avatar.php** and **General.php**. You got the idea. Next, inside

|  |  |
| --- | --- |
| 1 | app/Providers |

create new file called

|  |  |
| --- | --- |
| 1 | HelperServiceProvider.php |

with the following content

|  |  |  |
| --- | --- | --- |
| 01 | <?php namespace Darwin\Providers; | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | use Illuminate\Support\ServiceProvider; | |
| 04 |  |

|  |  |  |
| --- | --- | --- |
| 05 | class HelperServiceProvider extends ServiceProvider { | |
| 06 |  |

|  |  |
| --- | --- |
| 07 | /\*\* |
| 08 | \* Bootstrap the application services. | |

|  |  |
| --- | --- |
| 09 | \* |
| 10 | \* @return void | |

|  |  |
| --- | --- |
| 11 | \*/ |
| 12 | public function boot() | |

|  |  |
| --- | --- |
| 13 | { |
| 14 | // | |

|  |  |  |
| --- | --- | --- |
| 15 | } | |
| 16 |  |

|  |  |
| --- | --- |
| 17 | /\*\* |
| 18 | \* Register the application services. | |

|  |  |
| --- | --- |
| 19 | \* |
| 20 | \* @return void | |

|  |  |
| --- | --- |
| 21 | \*/ |
| 22 | public function register() | |

|  |  |
| --- | --- |
| 23 | { |
| 24 | foreach (glob(app\_path().'/Helpers/\*.php') as $filename){ | |

|  |  |  |
| --- | --- | --- |
| 25 | require\_once($filename); | |
| 26 | } |

|  |  |  |
| --- | --- | --- |
| 27 | } | |
| 28 | } |

Again, adjust the namespace.

Next let’s tell Laravel about this Service Provider. Open

|  |  |
| --- | --- |
| 1 | config/app.php |

inside **providers** array add

|  |  |
| --- | --- |
| 1 | 'Darwin\Providers\HelperServiceProvider', |

and inside **aliases** array add

|  |  |
| --- | --- |
| 1 | 'Text' => 'Darwin\Helpers\Text', |

Guess what? We’re done.

**Usage**

How to use defined helper function?

Inside any view you can now use

|  |  |
| --- | --- |
| 1 | <td><a href="{{ URL::route('article\_show', [$article->id]) }}" |
| 2 | title="{{ $article->title }}">{{ Text::shorten($article->title, 75) }}</a></td> | |

You just simply call a static function

|  |  |
| --- | --- |
| 1 | Alias::static\_function\_name(param1, param2... param n) |

How about controllers?

First tell Laravel where to look for the Text class. After the namespace definition add

|  |  |
| --- | --- |
| 1 | use Darwin\Helpers\Text; |

and now everywhere inside controller you can call **shorten** like

|  |  |
| --- | --- |
| 1 | Flash::success('Article "' . Text::shorten($article->title, 50) . '" has been successfully created.'); |

I hope that my solution will be useful in your project.