


19HRS 16MINS 47SECS

✕

Get All Our Books And Courses For \$9 (https://www.sitepoint.com/premium/l/join?ref_source=sitepoint&ref_medium=noticebar)

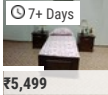
INSTANT DISCOUNT
UP TO ₹400 OFF*

7+ Days




₹5,499

7+ Days



7+ Days



Report Advertisement

✕

Become an Expert PHP Developer

Get All our PHP books & courses, plus SitePoint's entire library for **Just \$9/month.**



GET FULL ACCESS
([/Premium/L/Join-Php?
Ref_source=Sitepoint&Ref_medium=
Phppushdown&Ref_campaign=Creat
e-Laravel-Css-Minify-
Command&Ref_content=%22php%22
\)](https://www.sitepoint.com/premium/l/join-php?ref_source=sitepoint&ref_medium=phppushdown&ref_campaign=create-laravel-css-minify-command&ref_content=%22php%22))

Php (<https://www.sitepoint.com/php/>) > June 11, 2014 > By [Younes Rafie \(https://www.sitepoint.com/author/vrafie/\)](https://www.sitepoint.com/author/vrafie/)

How to Create a Laravel CSS-Minify Command

In this article you'll learn how to use Laravel's Artisan command line tool, and how to create a customized command. Note that you need to be familiar with the Laravel framework to get the most of this article.

What are we building

In this tutorial we're going to build a command to minify our css assets, which will be used like this:

```
cssmin 'output_path' 'file1'...'fileN' --comments --concat
```

output_path: (required) path to save the minified files, (`style.css` -> `style.min.css`).

file1 ... fileN: (required) list of files to minify.

--comments: (optional) add this option to keep comments.

--concat: (optional) concatenate the minified files into one file called `all.min.css`.

What is a Laravel Command

Artisan is the name of the command line utility in Laravel. It comes with a set of predefined commands, which you can list with `php artisan list`. If you want to show the help for a specific command, you can use `php artisan help command`.

Creating the CSS Minifier Command

To create an artisan command, you can use the `command:make` command. This command accepts one argument:

19HRS 16MINS 47SECS



name: the class name for the command.

Get All Our Books And Courses For \$9 (https://www.sitepoint.com/premium/l/join?ref_source=Sitepoint&ref_medium=Noticebar)

and three options:

--command: the name that should be typed to run the command.

--path: by default the commands are stored within the `app/commands` folder, however, you can change that with this option.

--namespace: you can use this option to namespace your set of commands, e.g. in the command `command:make`, the `make` command is under the `command` namespace.

Now, to create our command we will use `php artisan command:make CssMinCommand --command=cssmin` which will create a `CssMinCommand.php` file within our `app/commands` directory.

```
use Illuminate\Console\Command;
use Symfony\Component\Console\Input\InputOption;
use Symfony\Component\Console\Input\InputArgument;

class CssminCommand extends Command{
    protected $name = 'cssmin';
    protected $description = 'Command description.';

    public function __construct(){
        parent::__construct();
    }

    public function fire(){
        //
    }

    protected function getArguments(){
        return array(
            array('example', InputArgument::REQUIRED, 'An example argument.'),
        );
    }

    protected function getOptions(){
        return array(
            array('example', null, InputOption::VALUE_OPTIONAL, 'An example option.', null),
        );
    }
}
```

Our `CssMinCommand` class extends the `Illuminate\Console\Command` and overrides two methods (`getArguments`, `getOptions`).

getArguments: this function returns an array of arguments that should be passed to the command, (ex: the list of files that we pass to the `cssmin` command).

getOptions: returns a list of options or switches that you may pass to the command. (e.g. `--comments`).

Note: options may or may not have values, `--comments` is only a flag that returns `true` if it's passed to the command, whereas `--output='public/assets'` will return a value.

When your command is executed, the `fire` method is called, so this is where we need to put our command logic.

Registering the Command

if you try to run our command `php artisan cssmin 'args'` you'll get a `Command "cssmin" is not defined.`

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy \(/privacy-policy\)](#) Have questions? Please contact support@sitepoint.com

To register a command you need to add it to the `artisan.php` file.

(<mailto:support@sitepoint.com>) we're happy to help!

Understood

[\(https://www.sitepoint.com/\)](https://www.sitepoint.com/)
Artisan::add(new CssMinCommand);

//or through the container
Artisan::add(App::make("CssMinCommand"));

19HRS 16MINS 47SECS

Get All Our Books And Courses For \$9 (https://www.sitepoint.com/premium/l/join?ref_source=sitepoint&ref_medium=noticebar)

X

If you don't want to put your commands in the `artisan.php` file, you can create a separate file and include it, or if you're creating a package you can register them in your [Service Provider](http://laravel.com/docs/packages#service-providers) (<http://laravel.com/docs/packages#service-providers>).

Arguments

In our `getArguments` method we will define our `output` and `files`.
To define an argument, we need to pass an array of values:

```
array( 'name', 'mode', 'description', 'defaultValue' )
```

name: key name to be used when retrieving arguments.
mode: can have one of the three options:

- InputArgument::REQUIRED:** argument is required.
 - InputArgument::OPTIONAL:** argument is optional.
 - InputArgument::IS_ARRAY:** argument accepts multiple values (ex: `file1...fileN`).
- However, you can combine them like `InputArgument::IS_ARRAY | InputArgument::REQUIRED` (argument is required and must be an array).

description: useful when printing the command help.
defaultValue: if argument was not provided.

So our `getArguments` method will be:

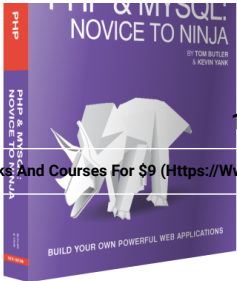
```
protected function getArguments(){  
    return array(  
        array(  
            'output',  
            InputArgument::REQUIRED,  
            'Path to output directory'  
        ),  
        array(  
            'files',  
            InputArgument::IS_ARRAY | InputArgument::OPTIONAL ,  
            "List of css files to minify"  
        ),  
    );  
}
```

Note: when using the `IS_ARRAY` argument it should be the last one on the returned arguments array. (obviously).

Options

Our `cssmin` command will only have two options. To define an option we pass an array:

```
array('name', 'shortcut', 'mode', 'description', 'defaultValue')
```



needed to build a fully-functional database-driven website using PHP & MySQL.

19HRS 16MINS 47SECS

Enter your email



Get All Our Books And Courses For \$9 (https://www.sitepoint.com/premium/l/join?ref_source=sitepoint&ref_medium=noticebar)

GET YOUR FREE BOOK NOW

You'll be subscribed to the back-end newsletter

name: the name of your option (ex: **comments**).

shortcut: a shorter version of your option (ex: **--verbose** and **-v**).

mode: can be one of the four options (**InputOption::VALUE_IS_ARRAY**, **InputOption::VALUE_OPTIONAL**, **InputOption::VALUE_REQUIRED**, **InputOption::VALUE_NONE**), the first three values are similar to the arguments.

VALUE_NONE: indicates that the option is a boolean flag (ex: **--verbose**).

description: useful when printing the command help.

defaultValue: if option value was not provided.

So our `getOptions` method will be:

```
protected function getOptions(){
    return array(
        array('comments', 'c', InputOption::VALUE_NONE, 'Don\'t strip comments' , null),
        array('concat', null, InputOption::VALUE_NONE, 'Concat the minified result to one file' , null),
    );
}
```

Running the Command


When our `fire` method is called we need to gather our arguments and options. We can make a separate function to do that for us:

```
private function init(){
    // retrun an array
    $this->files = $this->argument('files');
    // return a string
    $this->output_path = $this->argument('output');
    // return true if passed, otherwise false
    $this->comments = $this->option('comments');
    // return true if passed, otherwise false
    $this->concat = $this->option('concat');
}
```

The `argument` and `option` methods take a key as an argument and return the appropriate value.

To keep our example clean and simple we will use this [simple function \(https://github.com/GaryJones/Simple-PHP-CSS-Minification/\)](https://github.com/GaryJones/Simple-PHP-CSS-Minification/) with a small modification for the minification process.

(<https://www.sitepoint.com/>)

private function minify(\$css, \$comments){

// Normalize whitespace

\$css = preg_replace('/\s+/', ' ', \$css);

19HRS 16MINS 47SECS



// Remove comment blocks [Get All Our Books And Courses For \\$9 \(https://www.sitepoint.com/premium/l/join?ref_source=Sitepoint&ref_medium=Noticebar\)](https://www.sitepoint.com/premium/l/join?ref_source=Sitepoint&ref_medium=Noticebar)

if(!\$comments){

\$css = preg_replace('/\/*\^[^\!](.*?)*/', '', \$css);

}//if

// Remove ; before }

\$css = preg_replace('/;(?!s*)/', '', \$css);

// Remove space after , : ; { } */ >

\$css = preg_replace('/(,|:|;|\{|\}|*|\/|>)/', '\$1', \$css);

// Remove space before , ; { } () >

\$css = preg_replace('/(,|:|;|\{|\}|\(|\)|>)/', '\$1', \$css);

// Strips leading 0 on decimal values (converts 0.5px into .5px)

\$css = preg_replace('/(:|)0\.[0-9]+(%|em|ex|px|in|cm|mm|pt|pc)/i', '\${1}.\${2}\${3}', \$css);

// Strips units if value is 0 (converts 0px to 0)

\$css = preg_replace('/(:|)(\.?)0(%|em|ex|px|in|cm|mm|pt|pc)/i', '\${1}0', \$css);

// Converts all zeros value into short-hand

\$css = preg_replace('/0 0 0 0/', '0', \$css);

// Shortern 6-character hex color codes to 3-character where possible

\$css = preg_replace('/#([a-f0-9])\1([a-f0-9])\2([a-f0-9])\3/i', '#\1\2\3', \$css);

return trim(\$css);

}//minify

Now to process our arguments (files) we're going to make a separate method to do the job.

[\(https://www.sitepoint.com/\)](https://www.sitepoint.com/)

Finally, our `fire` method will only call the two methods:

```
public function fire(){
    $this->init();
    $this->processFiles();
}
```

Tip: You can also run an external command using the `call` method.

```
$this->call('command:name', array('argument' => 'foo', '--option' => 'bar'));
```

To test our command, we're going to copy some css files into our **public/css** directory, and then run the command.

```
php artisan cssmin 'public/css' 'public/css/style.css' 'public/css/responsive.css'
```

```
php artisan cssmin 'public/css' 'public/css/style.css' 'public/css/responsive.css' --comments --concat
```

The first command will create two files (`style.min.css`, `responsive.min.css`) on the `public/css` directory.

Because we used the `--comments` and `--concat` flags, we're going to get a file called `all.min.css` containing the two files with comments left.

Our command is not very descriptive and doesn't give any messages or notifications!

Improving the Command

Improving the Command

We have worked to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy \(/privacy-policy\)](#). Have questions? Please contact support@sitepoint.com.

Before we continue, on the final GitHub repository I will create a new tag for our command so you can switch and test each one.

(<mailto:support@sitepoint.com>) we're happy to help.

To make the command a little verbose, Laravel provides us with some output functions:

Understood

[\(https://www.sitepoint.com/\)](https://www.sitepoint.com/)

```
$th ->line("This is a simple line message");

$this->info("This is an info");

$this->comment("This is a comment");

$this->question("This is a question");

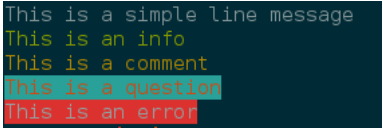
$this->error("This is an error")
```

19HRS 16MINS 47SECS

[Get All Our Books And Courses For \\$9 \(https://www.sitepoint.com/premium/l/join?ref_source=sitepoint&ref_medium=noticebar\)](#)

X

This will output:



Beside just displaying messages, you can ask the user for information, ex:

```
$confirm = $this->confirm("'style.min.css' already exists, overwrite?", false);

$filename = $this->ask("Save the file as?", 'all.min.css');

$choice = $this->choice(
    'Please select a level of minification:',
    [ 'minify all', 'leave comments' ],
    'default value'
);

$password = $this->secret('Type your password to confirm:');
```

The **confirm** method takes two arguments, a question message and a default value if the user type something different than y/n.

The **ask** method will ask the user for an input instead of just y/n, and if it's left empty, the default value is returned.

The **choice** method will give the user a numbered list to choose from, and if it's left empty, the default value is returned.

The **secret** method will prompt the user with a question and hide the typing, but the user input will be returned.

In fact, Laravel is just making [Symfony's Console API \(http://symfony.com/doc/current/components/console/helpers/dialoghelper.html\)](http://symfony.com/doc/current/components/console/helpers/dialoghelper.html) simpler and more verbose, and there is so much more if you want dig in.

Let's make our command more verbose and keep the user updated about the performed tasks.

```
private function processFiles(){
```

```
foreach ( $this->files as $file ) {
```

```
$this->info("Loading file");
```

```
//read file content
```

```
$file_content = file_get_contents( $file );
```

```
$this->info("minifying");
```

```
//minify CSS and add it to the result array
```

```
$css_result[] = $this->minify( $file_content, $this->comments );
```

```
//foreach
```

```
if( $this->concat ){
```

```
$this->comment("Concatenating into one file");
```

```
$css_concat = implode( PHP_EOL, $css_result );
```

```
$this->info("Saving to '{$this->output_path}/all.min.css'");
```

```
file_put_contents($this->output_path . '/all.min.css', $css_concat);
```

```
//if
```

```
else{
```

```
foreach ($css_result as $key => $css) {
```

```
//remove '.css' to add '.min.css'
```

```
$filename = basename( $this->files[$key], '.css' ) . '.min.css';
```

```
$this->comment("Saving '{$filename}'");
```

```
file_put_contents($this->output_path . '/' . $filename, $css);
```

```
    } //for
```

```
//else
```

```
//processFiles
```

```
$ art cssmin 'public' 'public/css/style.css' 'public/css/responsive.css' --comments
'public/css/style.css'
Loading file
minifying
'public/css/responsive.css'
Loading file
minifying
Saving 'style.min.css'
Saving 'responsive.min.css'
```

When creating an application, we are used to dumping the list of available routes (`php artisan routes`).

Domain	URI	Name	Action	Before Filters	After Filters
	GET /		Closure	auth	
	GET post/{id}	post	PostsController@show	auth	
	GET login	login.index	LoginController@index		
	POST login	login.store	LoginController@store		
	DELETE login/{login}	login.destroy	LoginController@destroy		
	GET signup	signup.index	SignupController@index		
	POST signup	signup.store	SignupController@store		
	GET upgrade	upgrade.index	UpgradeController@index		
	POST upgrade	upgrade.store	UpgradeController@store		

We'll see next how we can use some Symfony Console Helpers.

<mailto:support@sitepoint.com>), we're happy to help!

Understood

To illustrate the `ProgressBar` Helpers we will use the [Progress Helper \(http://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/06/1402500040command_progressbar.png\)](http://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/06/1402500040command_progressbar.png) to keep the user updated about the job progress.

At the end of our `init` method we will require a progress from the `HelperSet` `ProgressBar` `19 HRS 16 MINS 47 SECS`



```
Get All Our Books And Courses For $9 (https://www.sitepoint.com/premium/L/Join?Ref_source=Sitepoint&Ref_medium=Noticebar)

$this->progressbar = $this->getHelperSet()->get('progress');
$this->progressbar->start($this->output, count($this->files) );
```

The `start` method accepts two arguments, `$this->output` is a `ConsoleOutput` instance from the Symfony Console. The second argument is the maximum number of steps.

Every time we process a file in our `processFiles` method we will advance the progress bar by one step, and when the job is done we will end the progress bar and print a notification message.

```
private function processFiles(){
    $css_result = [];

    foreach ( $this->files as $file ) {
        //read file content
        $file_content = file_get_contents( $file );
        //minify CSS and add it to the result array
        $css_result[] = $this->minify( $file_content, $this->comments );
        // sleep for one second to see the effect
        //sleep(1);
        $this->progressbar->advance();
    }

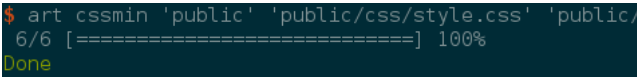
    if( $this->concat ){
        $css_concat = implode( PHP_EOL, $css_result );
        file_put_contents($this->output_path . 'all.min.css', $css_concat);
    }

    else{
        foreach ( $css_result as $key => $css ) {
            //remove '.css' to add '.min.css'
            $filename = basename( $this->files[$key], '.css' ) . '.min.css';

            file_put_contents($this->output_path . '/' . $filename, $css);
        }
    }

    $this->progressbar->finish();
    $this->info('Done');
}
```

You can try the command with multiple files or uncomment the `sleep` function line to see a live effect.




Note: This version will be tagged as `v3` on the final repository.

Conclusion

In this article we've learned how create and extend Laravel commands. Laravel has a lot of built-in commands that you can explore, and you can also check our [final repository \(https://github.com/sitepoint-examples/cssmin-laravelCommand\)](https://github.com/sitepoint-examples/cssmin-laravelCommand) on GitHub to test the final result. Questions? Comments? Would you like to see more Artisan Command tutorials? Let us know!

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legals\)](#) and [Privacy Policy \(/privacy-policy\)](#) Have questions? Please contact support@sitepoint.com

<mailto:support@sitepoint.com> we're happy to help!



Meet the author

Understood

Younes is a freelance web developer, technical writer and a blogger from Morocco. He's worked with JAVA, J2EE, JavaScript, etc., but his language of choice is PHP. You can learn more about him on his website.

Get All Our Books And Courses For \$9 (https://www.sitepoint.com/premium/L/Join?Ref_source=Sitepoint&Ref_medium=Noticebar)

Stuff We Do

- [Premium \(/premium/\)](#)
- [Versioning \(/versioning/\)](#)
- [Forums \(/community/\)](#)
- [References \(/html-css/css/\)](#)

About

- [Our Story \(/about-us/\)](#)
- [Press Room \(/press/\)](#)






Contact

- Contact Us (/contact-us/)
- FAQ (<https://sitepoint.zendesk.com/hc/en-us>)
- Write for Us (/write-for-us/)
- Advertise (/advertise/)

Legals

- [Terms of Use \(/legals/\)](#)
- [Privacy Policy \(/privacy-policy/\)](#)

Connect

 (<https://www.facebook.com/sitepoint>)
  (<http://twitter.com/sitepointdotcom>)
  ([/versioning/](#))
  (<https://www.sitepoint.com/feed/>)
  (<https://plus.google.com/+sitepoint>)

© 2000 – 2018 SitePoint Pty. Ltd.

Understood

We use cookies to provide you the best possible experience of SitePoint. Want to know more? Read our [Terms of Service \(/legal\)](#) and [Privacy Policy \(/privacy-policy\)](#) Have questions? Please contact support@sitepoint.com (<mailto:support@sitepoint.com>) we're happy to help!

Understood

