

Learn to code — free 3,000-hour curriculum

DECEMBER 8, 2020 / #LARAVEL

How to Use Facades in Laravel



Sule-Balogun Olanrewaju Ganiu

Facades are one of the key things you should understand when learning Laravel.

It took me a considerable amount of time to figure out how facades work, and I'm writing this to help anyone who is having trouble wrapping their heads around the concept.

In this article we'll cover what facades are, how they're used in Laravel, how you can build your own simple facade, and more.

What is a facade? And what is a wrapper?

A facade in Laravel is a wrapper around a non-static function that turns it into a static function.

The word "wrapper" can also be used when describing design patterns. Wrapping an object to provide a simplified interface to it is often described as the "facade" pattern.

Learn to code — free 3,000-hour curriculum

what static and non-static functions are in PHP.

Static Methods

In Static methods we're not required to create an instance of a class to reference it. Static methods use double colons (::) when accessing properties or methods of a class:

```
<?php
class Calc {
    const GOLDEN_RATIO = '1.618';
}

echo Calc::GOLDEN_RATIO; //1.618
```

Reserved keywords like `self` , `static` and `parents` is used to reference properties or methods within a class:

```
<?php
class backend {
    private const language = "php";
    public static function language() {
        echo self::language;
    }
}

backend::language(); //php
```

Non-static Methods

Learn to code — free 3,000-hour curriculum

```
<?php
class backend{

    public function language($name){

        echo $name;
    }

}

$test = new backend; //creating an instance of the class

$test->language('php'); //php
```

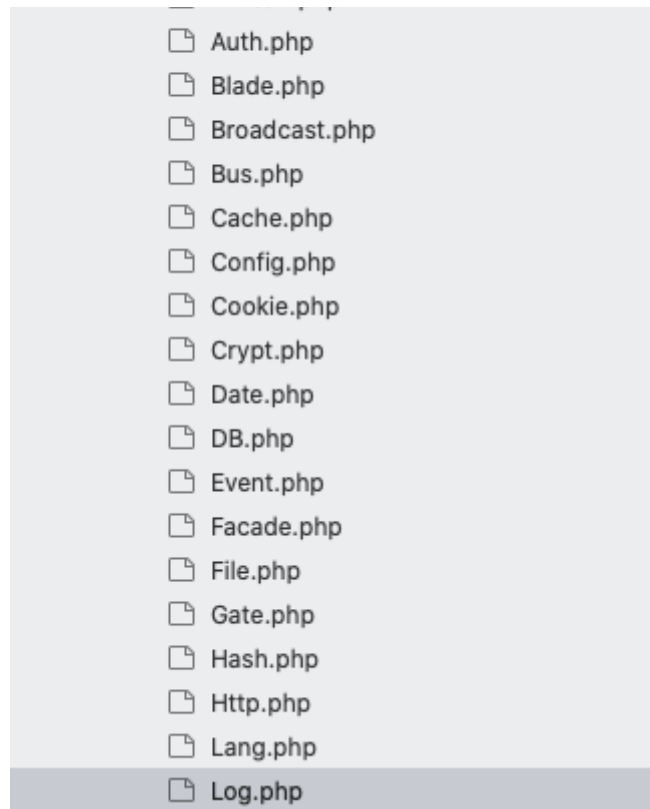
Now that we've gone over static and non-static methods, we can dive deeper into facades in Laravel.

Laravel facades

In the vendors > laravel > framework > src > illuminate > support > Facades directory, there's a list of files which are the various facades that ship with Laravel by default.

Here's a screenshot of what the directory structure actually looks like in our editor:

Learn to code — free 3,000-hour curriculum



Let's use the working code from `Log.php` to examine facades in more detail – the same explanation should apply to all facades in any Laravel application.

Laravel's Log facade

Here's the code for Laravel's `Log` facade:

```
<?php

namespace Illuminate\Support\Facades;

class Log extends Facade
{
    /**
     * Get the registered name of the component.
     *
     * @return string
     */
}
```

Learn to code — free 3,000-hour curriculum

```
}
```

`Log` is a class that extends the base facade which is from the namespace above.

Within the `Log` class we have a protected access modifier, `getFacadeAccessor`, and what that method does is it just returns `log`.

The name of this facade, `log`, is being returned so we can access the named facade anywhere within the Laravel application without initializing it. So we can do something like `Log::info('hello there');` anywhere really easily.

As you can see, facades make code easier to read, more organized, and make testing 10 times easier.

Since learning about `Log` from one of my co-workers, it's been my favorite debugging tool.

How to create a facade in Laravel

In this section we'll implement our own facade. The main objective here is to help learners understand how Laravel facades work.

We'll do this by creating a `StudentFacade` which will extend properties from a base `Facade` that returns a `name` property after it has been resolved. This `name` property will be of type `string` and it will be returned each time we instantiate the class as shown below:

Learn to code — free 3,000-hour curriculum



Curious how we will achieve this? Follow along as I'll be walking you through the steps.

We won't be creating our facade using the normal Laravel convention where we have a `.php` file in `app > facade` and then another in the `providers` before we end up registering it in the `config > app`.

Instead, we will make do with the `web.php` inside the `routes` for this illustration since we're just trying to see how facades work under the hood in a typical Laravel application.

First, let's start with this in `web.php` :

```
<?php
class Student{
    public function students(){
        return 'Sean';
    }
}

app()->bind('student', function(){
    return new Student;
});
```


We've created a class `Student` , and inside it we have a non-static `students` method that returns an array of students.

Learn to code — free 3,000-hour curriculum

Next, let's create a base `Facade` class still within the same `web.php` :

```
class Facade{
    public static function __callStatic($name, $args){
        return app()->make(static::getFacadeAccessor())->$name()
    }

    protected static function getFacadeAccessor(){
        //override take place
    }
}
```



Any facade we might create later will be extending the properties of this base facade.

Within the `Facade` class we have a `__callStatic` magic method that helps us resolve the `static::getFacadeAccessor()` from the container with `app()->make()` . And with those we're able to access the `$name` property.

```
class StudentFacade extends Facade {
    protected static function getFacadeAccessor(){
        return 'student';
    }
}
```

Here, `StudentFacade` inherits the properties of the base facade. Then we override `getFacadeAccessor()` and set the return value to be whatever we have each time we instantiate in the bind above `student` .

Learn to code — free 3,000-hour curriculum

When we try to call the facade which we created it returns "Sean" as expected. Now in the final step we have to put all these steps together:

```
<?php

class Student{
    public function students(){
        return 'Sean';
    }
}


app()->bind('student', function(){
    return new Student;
});

class Facade{
    public static function __callStatic($name, $args){
        return app()->make(static::getFacadeAccessor())->$name()
    }

    protected static function getFacadeAccessor(){
        //override take place
    }
}

class StudentFacade extends Facade {
    protected static function getFacadeAccessor(){
        return 'student';
    }
}

//log or die it to the output
dd(StudentFacade::students());
```



Learn to code — free 3,000-hour curriculum



Conclusion

I hope that by the end of this lesson you have been able to broaden your knowledge about facades work. If you have questions or wish to continue the conversation feel free to tweet at me.

References

[Laravel Beginner tutorial](#) - Bitfumes

[What is WRAPPER in programming, what does it help to do?](#) - Stackoverflow



Sule-Balogun Olanrewaju Ganiu

Experienced software engineer with a passion for developing innovative programs , well versed in technology and writing code to create systems that are reliable and user friendly.

If you read this far, tweet to the author to show them you care.

[Tweet a thanks](#)

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers.

Learn to code — free 3,000-hour curriculum

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charity organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can make a tax-deductible donation here.

Trending Guides

Python list .append()	C++ Operators
Python .zip() Function	Excel Formulas
What is a QA Engineer?	Break in Python
Python Print Same Line	Text Align in CSS
Rename Branches in Git	How to Minify CSS
What Does Coding Mean?	Python Split String
What is Data Analysis?	Python List insert()
How to Comment Out CSS	Merge Sort Algorithm
Double vs Float in C++	What is an SVG File?
Python Global Variables	RGB Colors Explained
How Vectors Work in C++	Square a Number in Python
JavaScript .setTimeout()	How to Lock Cells in Excel
Excel Keyboard Shortcuts	Python Delete Key from Dict
Excel Absolute Reference	Beginner Tech Jobs Examples
Insert Checkbox in Excel	Crow's Foot Notation

[Forum](#)

[Donate](#)

[Learn to code — free 3,000-hour curriculum](#)

[Code of Conduct](#)

[Privacy Policy](#)

[Terms of Service](#)

[Copyright Policy](#)