Quickstart

Are you a visual learner?

Master Livewire with our in-depth screencasts

Watch now

To begin your Livewire journey, we will create a simple "counter" component and render it in the browser. This example is a great way to experience Livewire for the first time as it demonstrates Livewire's *liveness* in the simplest way possible.

Prerequisites

Before we start, make sure you have the following installed:

- Laravel version 10 or later
- PHP version 8.1 or later

Install Livewire

From the root directory of your Laravel app, run the following Composer command:

```
composer require livewire/livewire
```

Make sure Alpine isn't already installed

> If the application you are using already has AlpineJS installed, you will need to remove it for Livewire to work properly; otherwise, Alpine will be loaded twice and Livewire won't function. For example, if you installed the Laravel Breeze "Blade with Alpine" starter kit, you will need to remove Alpine from `resources/js/app.js`.

Create a Livewire component

Livewire provides a convenient Artisan command to generate new components quickly. Run the following command to make a new `Counter` component:

```
php artisan make:livewire counter
```

This command will generate two new files in your project:

- `app/Livewire/Counter.php`
- `resources/views/livewire/counter.blade.php`

Writing the class

Open `app/Livewire/Counter.php` and replace its contents with the following:

```
<?php

namespace App\Livewire;

use Livewire\Component;

class Counter extends Component
{
    public $count = 1;

    public function increment()
    {
```

```
        $this->count++;
    }

    public function decrement()
    {
        $this->count--;
    }

    public function render()
    {
        return view('livewire.counter');
    }
}
```

Here's a brief explanation of the code above:

- `public $count = 1;` — Declares a public property named `$count` with an initial value of `1`.
- `public function increment()` — Declares a public method named `increment()` that increments the `$count` property each time it's called. Public methods like this can be triggered from the browser in a variety of ways, including when a user clicks a button.
- `public function render()` — Declares a `render()` method that returns a Blade view. This Blade view will contain the HTML template for our component.

Writing the view

Open the `resources/views/livewire/counter.blade.php` file and replace its content with the following:

```
<div>
    <h1>{{ $count }}</h1>

    <button wire:click="increment">+</button>

    <button wire:click="decrement">-</button>
</div>
```

This code will display the value of the `$count` property and two buttons that increment and decrement the `$count` property, respectively.

> **Livewire components MUST have a single root element** 
>
> In order for Livewire to work, components must have just **one** single element as its root. If multiple root elements are detected, an exception is thrown. It is recommended to use a `<div>` element as in the example. HTML comments count as separate elements and should be put inside the root element. When rendering **full-page components**, named slots for the layout file may be put outside the root element. These are removed before the component is rendered.

Register a route for the component

Open the `routes/web.php` file in your Laravel application and add the following code:

```
use App\Livewire\Counter;

Route::get('/counter', Counter::class);
```

Now, our *counter* component is assigned to the `/counter` route, so that when a user visits the `/counter` endpoint in your application, this component will be rendered by the browser.

Create a template layout

Before you can visit `/counter` in the browser, we need an HTML layout for our component to render inside. By default, Livewire will automatically look for a layout file named: `resources/views/components/layouts/app.blade.php`

You may create this file if it doesn't already exist by running the following command:

```
php artisan livewire:layout
```

This command will generate a file called `resources/views/components/layouts/app.blade.php` with the following contents:

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>{{ $title ?? 'Page Title' }}</title>
    </head>
    <body>
        {{ $slot }}
    </body>
</html>
```

The *counter* component will be rendered in place of the `$slot` variable in the template above.

You may have noticed there is no JavaScript or CSS assets provided by Livewire. That is because Livewire 3 and above automatically injects any frontend assets it needs.

Test it out

With our component class and templates in place, our component is ready to test!

Visit `/counter` in your browser, and you should see a number displayed on the screen with two buttons to increment and decrement the number.

After clicking one of the buttons, you will notice that the count updates in real time, without the page reloading. This is the magic of Livewire: dynamic frontend applications written entirely in PHP.

We've barely scratched the surface of what Livewire is capable of. Keep reading the documentation to see everything Livewire has to offer.

LIVEWIRE