Build a To-Do App in Laravel: Step-by-Step Livewire Example

Written by Bhadresh Kotadiya
Jul 29, 2024

Dolphin
Web Solution Pvt. Ltd.
Build A To-Do App In Laravel :
Step-By-Step Livewire Example
dolphinwebsolution.com
sales@dolphinwebsolution.com

Table of Contents

Hide

Summary :

Privacy - Terms

> In this article, we will explore how to create a dynamic TODO list application using Livewire 3 with Laravel. This tutorial is for all of you who are pro in these frameworks and also a complete beginner, we will be showing how building up a simple TODO list can turn so powerful with this simplistic Livewire + Laravel combo (Love). So, let's crack it down and walk through how to use Livewire 3 - the latest set of features as an example by making a live ToDoList app that will enhance your development skills for the web and give you underlying functionality upending antics on which future projects can be built.

Overview Of Laravel and Livewire

Laravel is a robust and stylish PHP framework created for building web applications with a clear and elegant syntax. Thanks to its expressive syntax and powerful tools, many developers love using it to craft top-notch web applications. This post demonstrates a new approach for achieving this with Livewire, billed as the Laravel way to build better dynamic interfaces using PHP instead of JavaScript.

Livewire 3, the latest version, introduces new features and improvements that make building interactive and dynamic applications even more straightforward. With Livewire, you can create dynamic components that update in real-time, offering a seamless user experience without writing a single line of JavaScript.

Content

Create a new Laravel Project using the below command:

composer create-project –prefer-dist laravel/laravel livewire-todo

cd livewire-todo

After successfully creating a new project, install Livewire:

composer require livewire/livewire

Create Migration and Model

php artisan make:model Todo -m

database \ migrations \ 2024_07_05_073256_create_todos_table.php

```
Schema::create('todos', function (Blueprint $table) {
  $table->id();
  $table->string('name');
  $table->boolean('completed')->default(false);
  $table->timestamps();
});
```

App \ Models \ Todo

```
class Todo extends Model{
use HasFactory;
protected $guarded = [];
}
```

Create the Livewire Component

php artisan make:livewire TodoList

This command will create two files:

1.TodoList.php(app/Http/Livewire).
2.todo-list.blade.php(resources/views/livewire).

1.TodoList.php

```
<?php
namespace App\Livewire;
use Livewire\Attributes\Rule;
use Livewire\Component;
use Livewire\WithPagination;
class TodoList extends Component
{
    use WithPagination;
    #[Rule('required|min:3|max:50')]
    public $name;
    #[Rule('required|min:3|max:50')]
    public $EditingNewName;
    public $EditingTodoID;
    public $search;
    public function create(){
        $this->validateOnly('name');
        Todo::create(['name' => $this->name]);
        $this->reset('name');
        $this->resetPage();
        session()->flash('success', 'Todo Created
Successfully.');
    }
    public function delete($id){
        try {
            Todo::findOrFail($id)->delete();
        } catch (\Throwable $th) {
            session()->flash('error', 'Failed to
delete todo!');
            Log::error($th->getMessage());
            return;
        }
    }

    public function edit($id){
        $this->EditingTodoID = $id;
        $this->EditingNewName = Todo::find($id)-
>name;
    }

    function toggle($id){
        $todo = Todo::find($id);
        $todo->completed = !$todo->completed;
        $todo->save();
    }

    public function update(){
        $this->validateOnly('EditingNewName');
        Todo::find($this->EditingTodoID)->update([
            'name' => $this->EditingNewName,
            'created_at' => now(),
        ]);
        $this->cancel();
    }

    public function cancel(){
        $this->reset('EditingTodoID',
'EditingNewName');
    }
    public function render(){
        return view('livewire.todo-list', [
            'todos' => Todo::latest()->where('name',
'like', "%{$this->search}%")->paginate(3),
        ]);
    }
}
```

2.todo-list.blade.php

```
<div>
    @if (session('error'))
        <div class="bg-red-100 border-t-4 border-red-
500 rounded-b text-red-900 px-4 py-3 shadow-md"
role="alert">
            <div class="flex">
                <div class="py-1"><svg class="fill-
current h-6 w-6 text-red-500 mr-4"
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20
20">
                        <path d="M2.93 17.07A10 10 0
1 1 17.07 2.93 10 10 0 0 1 2.93 17.07zm12.73-1.41A8 8
0 1 0 4.34 4.34a8 8 0 0 0 11.32 11.32zM9 11V9h2v6H9v-
4zm0-6h2v2H9V5z"></path>
                    </svg></div>
                <div>
                    <p class="font-bold">Error</p>
                    <p class="text-sm">{{
session('error') }}</p>
                </div>
            </div>
        </div>
    @endif
    @include('includes.create-todo-list')
    @include('includes.search-box')
    <div>
        @foreach ($todos as $todo)
            @include('includes.todo-card', ['todo' =>
$todo])
        @endforeach
        <div class="my-3">
            <!-- Pagination goes here -->
            {{ $todos->links() }}
        </div>
    </div>
</div>
```

For the main Livewire component, we're going to create a subview for the TODO list items so that our file is kept
clean and organized. This is the responsible subview that will render an individual TODO item.

First, Create a New Blade File **(todo-item.blade.php)** inside the resources/views/includes directory:

resources/views/includes/create-todo-list.blade.php

```
<div class="container content py-6 mx-auto create-
todo">
    <div class="mx-auto div">
```

```
        <div class="hover:shadow p-5 bg-white border-
blue-500 border-t-2">
            <div class="flex">
                <h2 class="text-lg text-gray-800 mb-
4">Create New Todo</h2>
            </div>
            <div>
                <form>
                    <div class="mb-5">
                        <label for="title"
class="block mb-3 text-sm font-medium text-gray-900
dark:text-white">*
                            Todo </label>
                        <input wire:model="name"
type="text" id="title" placeholder=".." class="bg-
gray-100  text-gray-800 text-sm rounded block w-full
p-3">
                        @error('name')
                            <span class="text-red-500
text-xs mt-4 block ">{{ $message }}</span>
                        @enderror
                    </div>
                    <button
wire:click.prevent="create" type="submit" class="px-4
py-3 bg-blue-500 text-white rounded hover:bg-blue-
600">Create
                        +</button>
                    @if (session('success'))
                        <span class="text-green-600
text-xs">{{ session('success') }}</span>
                    @endif
                </form>
            </div>
        </div>
    </div>
</div>
```

resources/views/includes/search-box.blade.php

```
<div id="search-box" class="search-box flex flex-col
items-center px-2 my-4 justify-center">
    <div class="flex justify-center items-center
search">
        <svg xmlns="http://www.w3.org/2000/svg"
fill="none" viewBox="0 0 24 24" stroke-width="1.5"
stroke="currentColor" class="w-6 h-6">
            <path stroke-linecap="round" stroke-
linejoin="round" d="M21 21l-5.197-5.197m0 0A7.5 7.5 0
105.196 5.196a7.5 7.5 0 0010.607 10.607z"></path>
        </svg>
        <input
wire:model.live.debounce.500ms="search" type="text"
placeholder="search todo…." class="bg-gray-100 ml-2
rounded px-4 py-2 hover:bg-gray-100">
    </div>
</div>
```

resources/views/includes/todo-card.blade.php

```
<div wire:key="{{ $todo->id }}" class="todo mb-5 card
px-5 py-6 bg-white col-span-1 border-t-2 border-blue-
500 hover:shadow">
    <div class="flex justify-between space-x-2">
        <div class="flex items-center space-x-2
files">
            @if ($EditingTodoID == $todo->id)
            <input wire:model="EditingNewName"
type="text" class="bg-gray-100  text-gray-900 text-sm
rounded block w-full p-2.5">
            <div>
                @error('EditingNewName')
                <span class="text-red-500 text-xs
block mt-1">{{ $message }}</span>
                @enderror
            </div>
            @else
            <h3 class="text-lg text-semibold text-
gray-800">{{ $todo->name }}</h3>
            @endif
        </div>
        <div class="flex items-center space-x-2">
            <!-- Edit button -->
            <button wire:click="edit({{ $todo->id
}})" class="text-sm text-teal-500 font-semibold
rounded hover:text-teal-800">
                <svg
xmlns="http://www.w3.org/2000/svg" fill="none"
viewBox="0 0 24 24" stroke-width="1.5"
stroke="currentColor" class="w-4 h-4">
                    <path stroke-linecap="round"
stroke-linejoin="round" d="M16.862 4.487l1.687-
1.688a1.875 1.875 0 112.652 2.652L10.582 16.07a4.5
4.5 0 01-1.897 1.13L6 18l.8-2.685a4.5 4.5 0 011.13-
1.897l8.932-8.931zm0 0L19.5 7.125M18 14v4.75A2.25
2.25 0 0115.75 21H5.25A2.25 2.25 0 013
18.75V8.25A2.25 2.25 0 015.25 6H10"></path>
                </svg>
            </button>
            <!-- Delete button -->
            <button wire:click="delete({{ $todo->id
}})" class="text-sm text-red-500 font-semibold
rounded hover:text-teal-800 mr-1">
                <svg
xmlns="http://www.w3.org/2000/svg" fill="none"
viewBox="0 0 24 24" stroke-width="1.5"
stroke="currentColor" class="w-4 h-4">
                    <path stroke-linecap="round"
stroke-linejoin="round" d="M14.74 9l-.346 9m-4.788
0L9.26 9m9.968-3.21c.342.052.682.107 1.022.166m-
1.022-.165L18.16 19.673a2.25 2.25 0 01-2.244
2.077H8.084a2.25 2.25 0
                        01-2.244-2.077L4.772
5.79m14.456 0a48.108 48.108 0 00-3.478-.397m-12
.562c.34-.059.68-.114 1.022-.165m0 0a48.11 48.11 0
013.478-.397m7.5 0v-.916c0-1.18-.91-2.164-2.09-
2.201a51.964 51.964 0 00-3.32 0c-1.18.037-2.09 1.022-
2.09 2.201v.916m7.5 0a48.667 48.667 0 00-7.5 0">
</path>
                </svg>
            </button>
        </div>
    </div>
    <span class="text-xs text-gray-500"> {{ $todo-
>created_at }} </span>
    <div class="mt-3 text-xs text-gray-700">
        <div class="flex items-center justify-between
mt-3">
            @if ($EditingTodoID == $todo->id)
            <div class="flex items-center space-x-2">
                <button wire:click="update({{ $todo-
>id }})" class="px-4 py-2 bg-teal-500 text-white
font-semibold rounded hover:bg-teal-600">
                    Update
                </button>
                <button wire:click="cancel"
class="px-4 py-2 bg-red-500 text-white font-semibold
rounded hover:bg-red-600">
                    Cancel
                </button>
            </div>
            @else
            <div class="flex items-center space-x-2">
                <input type="checkbox"
id="toggleCheckbox{{ $todo->id }}"
wire:click="toggle({{ $todo->id }})" class="hidden"
{{="" $todo-="">completed ? 'checked' : '' }} />
                <label for="toggleCheckbox{{ $todo-
>id }}" class="cursor-pointer flex items-center
space-x-2">
                    @if ($todo->completed)
                    <i class="fas fa-check-square
text-green-500"></i>
                    <span class="ml-2 text-green-500
font-semibold">Completed</span>
                    @else
                    <i class="fas fa-square text-
gray-500"></i>
                    <span class="ml-2 text-gray-500
font-semibold">Mark as Completed</span>
                    @endif
                </label>
            </div>
            @endif
        </div>
    </div>
</div>
```

Finally, include the Livewire scripts in your **resources/views/welcome.blade.php:**

```
<!-- <!DOCTYPE html>
<html lang="en">
<head>
     <title>Todo App template</title>
    <script src="https://cdn.tailwindcss.com">
</script>
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0-beta3/css/all.min.css">
</head>
<body>
```

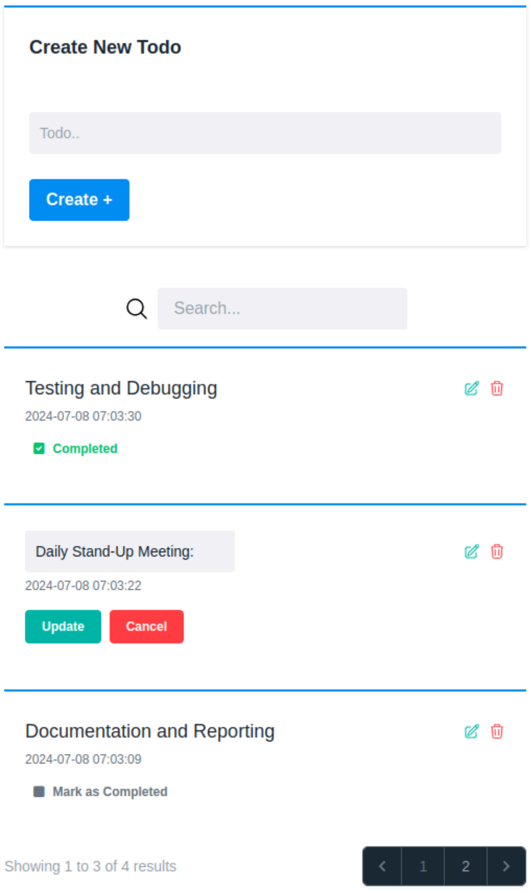
```
        <div id="content" class="mx-auto" style="max-
    width:510px;">
            @livewire('todo-list')
        </div>
    </body>
    </html> -->
```

Running the Application

Now, run the code to see that TODO list on your application screen. Type the following command and hit enter.

php artisan serve

Now go to your browser and visit http://localhost:8000 doing some magician 🙂



GitHub Repository

Here is a complete code for this project on my GitHub repository: GitHub Repository Link

Conclusion

And that’s it! In literally 5 minutes you built a dynamic TODO list app in Laravel + Livewire v3. Join us to build interactive applications using very little code, and see how simple and powerful Livewire is while building something practical There you have it, feel free to keep going on this project by continuing with such as storing tasks in a database or user authentication and more complex interactions.

Stay tuned for more tutorials and happy coding! 🙂

Dolphin Web Solution has dedicated and skilled Laravel developers. Are you looking for a Laravel expert? If yes, then without wasting a second, contact us and hire a Laravel developer. We ensure to provide the best and most proficient Laravel developers who can meet your requirements.