Livewire

Setup je Laravel macam biasa.. buat model dan buat migration file untuk create table yang terkait dengan model..

Then, baru install livewire

composer **require** livewire/livewire

then, kita boleh mula.. mula dengan wat livewire component

php artisan make:livewire posts

so kita dapati, dia create dua tempat..

app/Http/Livewire/Posts.php

resources/views/livewire/posts.blade.php

dekat file app/Http/Livewire/Posts.php **//file ni ibarat controller aaa untuk livewire**

sini kita akan tulis render(), resetInputFields(), store(), edit(), cancel(), update(), and delete().

Bahagian ni ada beberapa part..

First dekat luar function (global variable):

**public** $posts, $title, $body, $post\_id;

public $counter = 0;

public function render() //function utama untuk rendering page tu

{

$this->posts = Post::all(); //ni incase kit ana retrieve dari model

return view('livewire.posts');

}

\*tak perlu passing variable dah.. yang declare dekat global semua boleh direct access dkt livewire component.

**Route macam biasa – Cuma perlu letak tag panggil component livewire yang mana**

Next, default route kita dekat welcome kan? Welcome.blade.php.. so dalam case ni.. biar je, kita guna route default..

Replace code dalam Welcome.blade.php

<!DOCTYPE html>

<html>

<head>

    <title>Livewire</title>

    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" rel="stylesheet">

    @livewireStyles

</head>

<body>

    <div class="container">

        <!-- part ni panggil livewire view component -->

        @livewire('posts')

    </div>

    @livewireScripts

</body>

</html>

* Tengok ada panggil component posts………… @livewire(‘posts’)

So dekat livewire view pulak, yang digenerated tadi.. resources/views/livewire/posts.blade.php

<div>

    <h1 style="margin-top : 150px;">

        {{ $counter }}

    </h1>

    <button wire:click="tambahCounter" type="button" class="btn btn-dark">Add(+)</button>

</div>

\*kita boleh terus guna {{ $counter }} yang ada dekat global tadi..

Then, kita boleh guna tag wire:click sebagai listener yang untuk buat realtime ni… dan panggil function mana yang ada dalam posts.php tadi

**Form submit/v-bind tapi dalam livewire**

<input type="text" class="form-control" wire:model.defer="limit" aria-describedby="LimitHelp" placeholder="Enter Limit">

wire:model=’namafunction’ //ni nama function dalam livewire punya controller.

\*untuk setiap perubahan dekat input, dia akan update terus dekat livewire controller punya global variable.. ibarat onChange=”ubah variable” //tapi ubah variable to dekat server side..

wire:model.defer=’namafunction’ //ni pulak adalah untuk, tak sentiasa update.. just update global variable kat controller tu, once disubmit..

**Once dah declare variable dekat livewire controller tu.. as GLOBAL**

\*kita tak yah passing dah dekat view /livewire component.. boleh terus access je..

@foreach($history as $hist)

                ({{ $hist->bil }}) images has {{ $hist->semasa }}

        @endforeach

\*terus mcm biasa – mcm kt laravel biasa

**Keep on update/any changes on database update to render (realtime)**

<h1 style="margin-top : 150px;" wire:poll='pollingChanges'>

        {{ $counter }} <br>

    </h1>

\*hanya letak tag wire:poll=’namafunction’ jee…

Actually, letak kt memana pun tak apa, asal dkt function tu update necessary things.

Misalnya,

public function pollingChanges()

    {

        $this->counter = counter::select('count')->first()->count;

        $this->history = sejarah::latest()->limit(5)->get();

    }

\*So sini dia update 2 values… so as nature of livewire, setiap kali global variable dia berubah, dia render component yang perlu kan? So begitulah..

\*ada beberapa jenis wire:poll

wire:poll.keep-alive //running on background

wire:poll.visible //polling when element attached is visible

**Storage Path (Laravel)**

storage\_path('app/downloaded/' . $this->arr[$i] . '')

\*ni merujuk ke folder storage/app/downloaded

**Buat realtime search – easy gila**

public $search = '';

public function render()

{

return view('livewire.search-users', [

'users' => User::where('username', $this->search)->get(), //setiap kali search berubah dia rerender..

]);

}

Dekat blade buat gini je..

<div>

<input wire:model="search" type="text" placeholder="Search users..."/> //sini lak dia wat v-bind gitu..so data always sync ah

<ul>

@foreach($users as $user)

<li>{{ $user->username }}</li>

@endforeach

</ul>

</div>

**Waktu kita panggil livewire component guna @livewire tu.. sebenarnye bole passing parameter**

@livewire('show-post', ['post' => $post])

**Na receiving parameter mcm mana?**

Just declare je as global dekat livewire controller tu.

**updated function (lifecycle hooks)**

Runs after any update to the Livewire component's data (Using wire:model, not directly inside PHP)

**Keeping track of component in loop.. sama je mcm Vue.. gune key**

@foreach ($users as $user)

@livewire('user-profile', ['user' => $user], key($user->id)) //see ada key

@endforeach

**\*#Setiap component kan ada view dia sekali kan? (controller-view) untuk livewire.**

So macam mana kalau component lain nak communicate dengan component lain (as long as dalam page yang sama)?

boleh guna listener dan emits..

listener untuk dengar kalau ada sapa panggil event dia, dan run function yang perlu..

emits untuk trigger panggil event tersebut..

**start dengan emits (firing events).**

1.Kalau dari template nak panggil events:

<button wire:click="$emit('postAdded')">

2.Kalau dari component (controller) nak panggil events:

$this->emit('postAdded');

3.Kalau dari Global Javascript nak panggil events:

<script>

Livewire.emit('postAdded')

</script>

\*so macam aku cakap awal tadi, emits ni untuk panggil event tak kisah dari elements/component mana pun.. boleh punya communicate, as long as semua dalam page yang sama.

Nak passing parameter pun boleh (incase function ditrigger tu ada parameter)..

$this->emit(‘postAdded’, $post->id);

**Event listeners (untuk dengar emits panggil event mana satu)**

public $postCount;

protected $listeners = ['postAdded' => 'incrementPostCount']; //postAdded tun ama event di emits

//incrementPostCount tu nama function.. function mana na run kalau terima emits mcm tu

public function incrementPostCount()

{

$this->postCount = Post::count();

}

Kalau tak nak declare listener mcm tu.. bole buat dalam bentuk function..

protected function getListeners()

{

return ['postAdded' => 'incrementPostCount'];

}

\*so emit ni untuk panggil yang event dkt component yang elements dia setara dengan dia atau child kepada dia.. kalau parent, kene guna emitUp(‘’)

Ada lagi banyak pasal emit ni.. buka documentation.

**Loading states (toggling elements)**

<div wire:loading>

Processing Payment...

</div>

\*banyak lagi – toggling class/attributes/etc.

**Offline states (no internet display certain thing)**

<div wire:offline>

You are now offline.

</div>

\*pun sama ada class/attributes.

**Prefetch (penting gila)**

<button wire:click.prefetch="toggleContent">Show Content</button>

@if ($contentIsVisible)

<span>Some Content...</span>

@endif

* when the mouse enters the "Show Content" button, Livewire will fetch the result of the "toggleContent" action in the background.
* If the button is actually clicked, it will display the content on the page without sending another network request.