Xây dựng ứng dụng chat realtime bằng laravel + vuejs + socket + redis

Xây dựng ứng dụng chat realtime bằng laravel + vuejs + socket + redis

Laravel giúp dễ dàng xây dựng các ứng dụng hiện đại với các tương tác thời gian thực bằng cách cung cấp một 'event broadcasting system' cho phép các nhà phát triển chia sẻ cùng tên sự kiện giữa server và ứng dụng JavaScript phía client. Trong bài này tôi sẽ chia sẻ làm như thế nào để xây dựng một ứng dụng chat realtime sử dụng laravel, Socket.IO, Redis và Vuejs

I. Cài đặt

1. Khởi tạo project

Bật Terminal lên, chúng ta khởi tạo project bằng lệnh:

```
Laravel new Chat
```

hoặc:

```
composer create-project --prefer-dist laravel/laravel Chat`
```

Sau khi tạo xong, chúng ta di chuyển vào trong project vừa tao, rồi cài Redis như sau:

```
composer require predis/predis
```

2. Setup Vuejs

VÌ Laravel đã tích hợp sẵn Vuejs vào project của mình nên chỉ cần chạy:

```
 npm install
```

II. Thực hiện

Đầu tiên chúng ta tạo migration table messages:

```
    Schema::create('messages', function (Blueprint $table) {
            $table->increments('id');
            $table->text('message');
            $table->integer('user_id')->unsigned();
            $table->timestamps();
        });
```

Rồi chạy lệnh:

```
php artisan migrate
```

Sau khi chạy xong laravel sẽ có table messages và table users.

Để chat được, user cần phải đăng nhập vào hệ thống, ta chỉ cần chạy lệnh:

```
php artisan make:auth
```

Sau đó chúng ta thêm vào router như sau:

```
    //lấy tất cả các messages, và sẽ có form để chat
    Route::get('messages', 'MessageController@index');

    //insert chat content vào trong database
    Route::post('messages', 'MessageController@store');

    //lấy ra user hiện tại
    Route::get('current-user', 'UserController@currentUser');
```

Tiếp theo chúng ta tạo ra model, controller Message và User:

```
 php artisan make:model Messsage -c
```

Câu lệnh trên sẽ tạo ra Model Message và MessageController

Tạo ra UserController:

```
php artisan make:controller UserController
```

Model của Message như sau:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Message extends Model
{
        protected $fillable = ['message', 'user_id'];

    public function user() {
        return $this->belongsTo('App\User');
    }
}
```

Model của User sẽ như sau:

```
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
```

```
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    public function message() {
        return $this->hasMany('App\Message');
    }
}
```

Trong MessageController sẽ như sau:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Message;

class MessageController extends Controller
{

    public function index()
    {
        if ($messages = Redis::get('messages.all')) {
            return json_decode($messages);
        }
        $messages = App\Message::with('user')->get();
        Redis::set('messages.all', $messages);

        return view('welcome');
    }

    public function store()
    {
        $user = Auth::user();
        $message = App\Message::create(['message'=> request()->get('message'), 'user_id' => $user->id]);
        broadcast(new MessagePosted($message, $user))->toOthers();

        return $message;
    }

}
```

Sau đó chúng ta cần tạo một Event MessagePosted bằng lệnh:

```
php artisan make:event MessagePosted
```

Trong MessagePosted có nội dung như sau:

```
<?php

namespace App\Events;

use Illuminate\Broadcasting\Channel;
use Illuminate\Queue\SerializesModels;
use Illuminate\Broadcasting\PrivateChannel;
use Illuminate\Broadcasting\PresenceChannel;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
use App\Message;
use App\User;

class MessagePosted implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;


    public $message;
    public $user;

    /**
     * Create a new event instance.
     *
     * @return void
     */
    public function __construct(Message $message, User $user)
    {
        $this->message = $message;
        $this->user = $user;
    }

    /**
     * Get the channels the event should broadcast on.
     *
     * @return \Illuminate\Broadcasting\Channel|array
     */
    public function broadcastOn()
    {
        return new Channel('chatroom');
    }
}
```

Và đừng quên vào config/app.js phần `providers` bỏ comment dòng này đi nhé

```
    App\Providers\BroadcastServiceProvider::class,
```

Ta cần update lại file .env

```
BROADCAST_DRIVER=redis
CACHE_DRIVER=redis
```

```
SESSION_DRIVER=redis
SESSION_LIFETIME=120
QUEUE_DRIVER=redis
```

Phần backend như thế coi như xong. Tiếp theo chúng ta sẽ làm việc với Vuejs
Trước tiên ta cần cài VueX và Axios vào trong project của mình:

```
npm install vuex --save
npm install axios --save
```

Chúng ta tạo ra 2 component là: ChatLayout và ChatItem trong resources/js/components.

Trong file ChatLayout có nội dung như sau:

```
<template>
    <div class="chat-layout">
        <div class="chat-layout-container">
            <div class="user-count">
                <h3>User count: </h3>
            </div>
            <div class="title">
                <h1>Chatroom</h1>
            </div>
            <div class="list-messages">
                <div class="message" v-for="message in list_messages">
                    <chat-item :message="message"></chat-item>
                </div>
            </div>
            <div class="input-section">
                <input type="text" v-model="message" class="input-el" placeholder="Enter some mssage..." @keyup
                <button @click="sendMessage">Send</button>
            </div>
        </div>
    </div>
</template>

<script>
    import ChatItem from './ChatItem.vue'
    export default {
        components: {
            ChatItem
        },

        data() {
            return {
                message: '',
                list_messages: []
            }
        },

        created() {
            this.loadMessage()
            Echo.channel('chatroom')
            .listen('MessagePosted', (data) => {
                console.log(data)
                if (data.to_user.id == this.$root.currentUserLogin.id) {
```

```
                let message = data.message
                message.user = data.user
                this.list_messages.push(message)
            }
        })
    },

    methods: {
        loadMessage() {
            this.$store.dispatch('getMessages')
            .then(res => {
                this.list_messages = res.data
            })
        },
        sendMessage() {
            axios.post('/messages', {
                    message: this.message
                })
                .then(response => {
                    this.list_messages.push({
                        message: this.message,
                        created_at: new Date().toJSON().replace(/T|Z/gi, ''),
                        user: this.$root.currentUserLogin
                    })
                    this.message = ''
                })
                .catch(error => {
                    console.log(error)
                })
        }
    }
}
</script>

<style lang="scss" scoped>
    .chat-layout {
        border: solid 1px #ddd;
        border-radius: 3px;
        padding: 20px;
        .chat-layout-container {
            .user-count {
                float: right;
            }
            .list-messages {
                .message{
                    padding: 5px 0;
                }
            }
            .input-section {
                .input-el {
                    width: 100%;
                    filter: hue-rotate(45deg);
                    font-weight: bold;
                    background-color: transparent;
                    border: 0;
```

```
                    border-bottom: 1px solid #404040;
                    outline: none;
                    overflow: visible;
                    font-size: 100%;
                    line-height: 1.15;
                    &:focus {
                        border-bottom: 1px solid #e400ff;
                    }
                }
            }
        }
    }
</style>
```

Trong ChatItem có nội dung như sau:

```
<template>
    <div class="chat-item">
        <div class="chat-item-container">
            <span class="message-time">{{ list.created_at }}</span>
            <span class="user-name"">{{ list.user.name }}</span>
            <span class="message-text">{{ list.message }}</span>
        </div>
    </div>
</template>

<script>
    export default {
        props: {
            message: {
                type: Object
            }
        },

        data() {
            return {
                list: this.message
            }
        }
    }
</script>

<style lang="scss" scoped>
    .current-user {
        color: red;
    }
</style>
```

Chúng ta tạo folder: `resources/js/store` (đây là nơi t sử dụng VueX để quản lý state). Sau đó chúng ta tạo file: store.js bên trong `resources/js/store` có nội dung như sau:

```
import Vue from 'vue';
import Vuex from 'vuex';
import axios from 'axios';
Vue.use(Vuex);
export const store = new Vuex.Store({
```

```
    states: {

    },

    mutations: {

    },

    actions: {
        getMessages () {
            var start_time = new Date().getTime();
            const a = axios.get('/messages')
            .then(res => {
                var request_time = new Date().getTime() - start_time;
                console.log(request_time)
                return res
            })
            return a
        }
    }
})
```

Trong resources/js/app.js:

```
/**
 * First we will load all of this project's JavaScript dependencies which
 * includes Vue and other libraries. It is a great starting point when
 * building robust, powerful web applications using Vue and Laravel.
 */

require('./bootstrap');

window.Vue = require('vue');
import Vuex from 'vuex'

Vue.use(Vuex)

/**
 * The following block of code may be used to automatically register your
 * Vue components. It will recursively scan this directory for the Vue
 * components and automatically register them with their "basename".
 *
 * Eg. ./components/ExampleComponent.vue -> <example-component></example-component>
 */

Vue.component('chat-layout', require('./components/ChatLayout.vue'));
import { store } from './store/store.js';
import axios from 'axios';

// const files = require.context('./', true, /\.vue$/i)

// files.keys().map(key => {
//     return Vue.component(_.last(key.split('/')).split('.')[0], files(key))
```

```
// })

/**
 * Next, we will create a fresh Vue application instance and attach it to
 * the page. Then, you may begin adding components to this application
 * or customize the JavaScript scaffolding to fit your unique needs.
 */

const app = new Vue({
    el: '#app',
    data: {
        currentUserLogin: {}
    },
    store: store,
    created() {
        this.getCurrentUserLogin()
    },
    methods: {
        getCurrentUserLogin() {
            axios.get('/getUserLogin')
            .then(response => {
                this.currentUserLogin = response.data
            })
        }
    },
});
```

Đừng quên chạy `npm run dev` nhé

Và ở trong `Welcome.blade.php` ta thêm:

```
<div id="app">
    <chat-layout></chat-layout>
</div>
<script type="text/javascript" src="{{ asset('js/app.js') }}"></script>
```

Ok. Bây giờ chúng ta xử lý đến Realtime Chúng ta cần cài: `socket.io` và `laravel echo`:

```
npm install --save socket.io-client laravel-echo laravel-echo-server
```

Sau khi cài đặt xong ta vào file resources/assets/js/bootstrap.js, kéo xuống cuối bỏ comment và sửa lại như sau:

```
import Echo from "laravel-echo"

window.io = require('socket.io-client');

window.Echo = new Echo({
    broadcaster: 'socket.io',
    host: window.location.hostname + ':6001'
});
```

Tiếp theo chúng ta setup laravel-echo-server nhé. Ta chạy command:

```
laravel-echo-server init
```

Cứ yes và bạn nhớ chọn Ređis nhé!!!

Note: các bạn mở file laravel-echo-server.json vừa được tạo , sửa lại trường authHost cho đúng với địa chỉ app chúng ta đang chạy (nếu không đến khi làm private channel sẽ lỗi nhé các bạn):

```
    "authHost": "http://localhost:8000",
```

Ta mở terminal và chạy lệnh:

```
laravel-echo-server start
```

và

```
php artisan serve
```

Mở trình duyệt và Xem thành quả nhé!!!

Chúc các bạn thành công!!!

Nguồn: Laravel, codeburst, Viblo