Laravel Livewire

1. Objectifs

- Connaitre Laravel Livewire

2. Présentation

- ***Laravel Livewire: un framework full-stack pour Laravel***
- **Livewire** est un framework full-stack pour Laravel qui simplifie la création d'interfaces dynamiques, sans quitter le confort de Laravel.
- Laravel **Livewire** est une bibliothèque qui simplifie la création d'interfaces modernes, réactives et dynamiques en utilisant Laravel Blade comme langage de création de modèles.



- **Livewire** est un framework full-stack pour le framework Laravel qui simplifie la création d'interfaces dynamiques, sans quitter le confort de Laravel.
- Si vous utilisez **livewire** avec Laravel, vous ne vous souciez pas d'écrire du code jquery ajax, livewire vous aidera à écrire de manière très simple du code jquery ajax en utilisant PHP sans actualisation de page.La validation Laravel fonctionnera, le formulaire sera soumis, etc.
- Vous pouvez utiliser Livewire pour implémenter les fonctionnalités suivantes, sans recharger la page:
  - pagination
  - validation de formulaire
  - notifications
  - aperçu des téléchargements de fichiers

apcpedagogie.com

3. Que fait le Laravel Livewire?

- Livewire rend la sortie du composant initial avec la page (comme un Blade include), de cette façon, il est convivial pour le référencement.
- Lorsqu'une interaction se produit, Livewire envoie une requête AJAX au serveur avec les données mises à jour.
- Le serveur restitue le composant et répond avec le nouveau HTML.
- Livewire mute alors intelligemment DOM en fonction des choses qui ont changé.

4. Comment fonctionne Livewire?

- Livewire restitue la sortie du composant initial sur la page - oui, comme le modèle de lame typique rendu par serveur
- Lorsqu'une interaction se produit, Livewire envoie une requête AJAX au serveur avec des données mises à jour
- Le serveur restitue le composant et répond avec du HTML mis à jour
- Livewire mute alors intelligemment le DOM en fonction de ce qui a changé
- Comme vous pouvez le voir, c'est comme avoir votre frontend et votre backend au même endroit, sans avoir besoin de répéter la logique.

5. Quelques Concepts clés de Livewire

- **Activation de Livewire sur une page**
  - Pour que Livewire fonctionne sur une page, vous devez inclure les styles et scripts Livewire sur chaque page qui en a besoin.
  - Habituellement, ceux-ci iraient dans votre modèle de base. Vous feriez cela en utilisant **`@livewireStyles`** et **`@livewireScripts`**.

```
//app.blade.php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>@yield</title>
    @livewireStyles  <<==================
</head>
```

apcpedagogie.com

```
<body>
    @yield ( 'contenu' )
    @livewireScripts       <<====================
</body>
</html>
```

6. **Créer un Livewire component**

   - Vous créez un composant Livewire en exécutant la commande suivante: **`php artisan make:livewire NomComponent`**
   - La commande ci-dessous créera également deux nouveaux fichiers: `app\Http\Livewire\NomComponent.php` et un modèle de vue **`resources/views/livewire/nomVue.blade.php`**.

7. **Actions Livewire**

   - Tout comme vous pouvez lier les données du modèle de vue aux propriétés publiques des composants, vous pouvez également mapper les événements côté client aux méthodes de vos composants.
   - Par exemple, vous pouvez répondre aux événements de clic, aux événements de keyup et de keydown, etc. à l'aide des méthodes définies dans votre classe de composant.
   - **Exemple**

```
<button wire:click="doSomething">Do Something</button>
<input wire:keydown.enter="doSomething">
<form wire:submit.prevent="save">
    ...
    <button>Save</button>
</form>
```

8. **La validation des données**

   - Livewire rend la validation des données transparente.
   - Pour valider les données provenant d'une vue de modèle de formulaire, vous écririez une propriété **`$rules`** contenant vos règles de validation, comme vous le feriez dans Laravel.
   - Ensuite, vous appelez le dans la méthode effectuant la validation.**`$this→validate()`**

```
...
class CreatePost extends Component
```

apcpedagogie.com

```
{
    public $title, $body;
    public $success;
    protected $rules = [
        'title' => 'required|string|max:220',
        'body' => 'required'
    ];

    public function render()
    {
        return view('livewire.create-post')
            ->extends('layouts.app')
            ->section('content');
    }
    public function create(){
        $this->validate();
        Post::create([
            'title' => $this->title,
            'slug' => Str::slug($this->title),
            'body' => $this->body,
            'author_id' => auth()->id()
        ]);

        $this->success = true;
    }
}


// livewire/create-post
<div class="container">
    @if ($success)
        <div class="alert alert-success">
            Post has been created successfully
        </div>
    @endif
    <form wire:submit.prevent="create">
        <div class="form-group">
            <label for="Post title">Post title</label>
            <input wire:model="title" type="text" name="title" id="title" class="form-
control" placeholder="Title of the post">
            @error('title') <span class="error">{{ $message }}</span> @enderror
        </div>
        <div class="form-group">
```

apcpedagogie.com

```
            <label for="Post body">Post Body</label>
            <textarea name="body" id="body" placeholder="Body of post here..."
wire:model="body" class="form-control"></textarea>
            @error('body') <span class="error">{{ $message }}</span> @enderror
        </div>
        <div>
            <button class="btn btn-primary" type="submit">Publish</button>
        </div>
    </form>
</div>
```

Sommaire du cours Laravel

Source : https://apcpedagogie.com/laravel-livewire · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com