Laravel 8 Jetstream Livewire CRUD

[sommaire]

1. **Objectifs**

- Être capable de créer des opérations CRUD livewire avec jetstream dans une application laravel 8

- **Qu'est-ce que Jetstream?**

  - **Laravel Jetstream**, sortira avec Laravel 8 le 8 septembre 2020, est un tout nouveau package d'échafaudage d'application, qui comprend des fonctionnalités pratiques, et constitue un point de départ idéal pour un projet Laravel 8.
  - **Laravel Jetstream** est une nouvelle prise en charge d'application pour Laravel. Laravel Jetstream remplace l'interface utilisateur d'authentification Laravel disponible dans l'ancienne version de Laravel.
  - **Jetstream** comprend des fonctionnalités telles que :

  1. fonction de connexion et d'enregistrement
  2. vérification des e-mails
  3. authentification à deux facteurs
  4. gestion de session
  5. fournit le support de l'API

- Laravel Jetstream remplace l'interface d'authentification Laravel disponible dans l'ancienne version de Laravel.

- **Créer un nouveau projet Laravel8**

  - Dans la première étape, nous devons obtenir une nouvelle application de version laravel à l'aide de la commande ci-dessous. Alors ouvrons le terminal et exécutons la commande ci-dessous.

```
composer create-project --prefer-dist laravel/laravel livewire-jetstream-projet01
```

- Quand on installe Laravel, si on veut déjà un système d’authentification complet, on installe Jetstream qui va lui-même installer Livewire comme dépendance :

apcpedagogie.com

- ## Créer Jetstream Auth avec Livewire

  - Installer **jetstream** à l'aide de la commande ci-dessous, alors ouvrir le terminal et exécuter la commande ci-dessous:

```
composer require laravel/jetstream
```

- Maintenant, créer une **authentification** à l'aide de la commande ci-dessous, vous pouvez créer **une connexion** de base, une **inscription** et une **vérification par e-mail**. si vous souhaitez créer **une gestion d'équipe**, vous devez passer le **paramètre d'addition**. vous pouvez voir les commandes ci-dessous:

```
php artisan jetstream:install livewire
```

- Maintenant, installer le package **node js**:

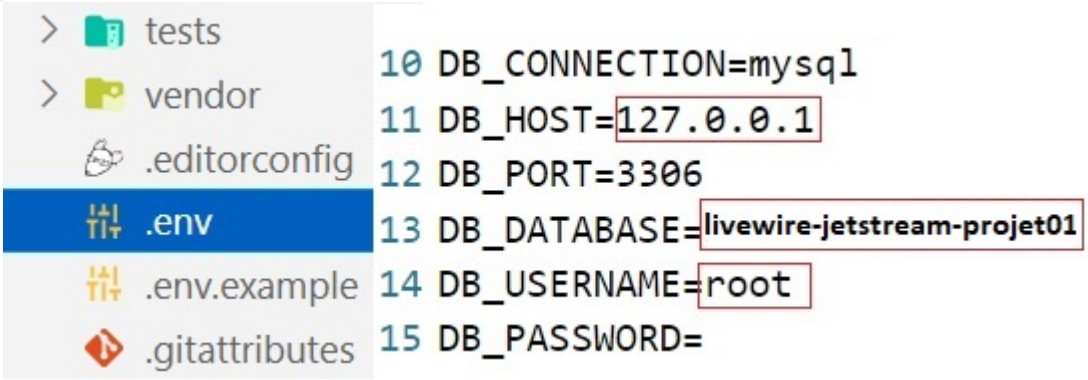
```
npm install
```

- Exécuter le package:

```
npm run dev
```

- ## Configurer la base de données

  - Pour configurer la base de données. Ouvrir le fichier "`.env`"
  - changer le nom de la base de données ("**livewire-jetstream-projet01**"), le nom d'utilisateur et le mot de passe.



apcpedagogie.com

- Si votre base de données est de type MySql5.6.XXX ou autre version antérieur, vous risquez d'obtenir un erreur en cours de migration, une solution (la plus simple mais pas la meilleure) consiste à modifier le fichier **database.php** dans le dossier config :
    - `'engine' => null,`
    - Devient :`'engine' => 'InnoDB ROW_FORMAT=DYNAMIC',`
    - Ou tout autre type de engine paramétré lors de l'install de MySql...

Créer une migration et un modèle

1. **Une migration**

- Dans cette étape, nous devons créer la table de migration pour les **blogs** à l'aide de la commande artisanale ci-dessous. Alors ouvrons le terminal et exécutons la commande:

```
php artisan make:migration create_blogs_table
```

- Après l'exécution de cette commande un fichier sera crée dans le chemin suivant: `database/migrations/2021_04_11_085510_create_blogs_table.php`
- Modifier la méthode up comme suit:

```
public function up()
    {
        Schema::create('blogs', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('body');
            $table->timestamps();
        });
    }
```

- Pour exécuter la migration pour un seul fichier, utiliser la commande suivante:`php artisan migrate --path=database/migrations/2021_04_11_085510_create_blogs_table.php`

- **Un model**

- Dans cette étape, nous devons créer lemodèle **Blog** à l'aide de la commande artisanale ci-dessous. Alors ouvrons le terminal et exécutons la commande:

```
php artisan make:model Blog
```

apcpedagogie.com

- Modifier le fichier "**app\Models\Blog.php**" comme suit:

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Blog extends Model
{
    use HasFactory;
    protected $fillable = [
        'title', 'body',
    ];
}
```

- ## Créer un composant (component) de blog

  - Dans cette étape, nous devons **créer un composant de blog livewire** à l'aide de la commande ci-dessous. Alors ouvrons le terminal et exécutons la commande ci-dessous:
    - `php artisan make:livewire blogs`
  - Maintenant, ils ont créé des fies sur les deux chemins:
    - **app/Http/Livewire/Blogs.php ressources/vues/livewire/blogs.blade.php**

- ## Mettre à jour le fichier de composant (component)

  - Ici, nous écrirons les méthodes `render()`, `create()`, `openModal()`, `closeModal()`, `resetInputFields()`, `store()`, `edit()` et `delete()` pour notre application CRUD.
  - Alors, mettons à jour le fichier comme suit:

apcpedagogie.com

```php
<?php
namespace App\Http\Livewire;
use Livewire\Component;
use App\Models\Blog;
class Blogs extends Component
{
    public $blogs, $title, $body,
           $blog_id;
    public $isOpen = 0;
    public function render()
    {
        $this->blogs = Blog::all();
        return view('livewire.blogs');
    }
    public function create()
    {
        $this->resetInputFields();
        $this->openModal();
    }
    public function openModal()
    {
        $this->isOpen = true;
    }
    public function closeModal()
    {
        $this->isOpen = false;
    }
    private function resetInputFields(){
        $this->title = '';
        $this->body = '';
        $this->blog_id = '';
    }
    public function store()
    {
        $this->validate([
            'title' => 'required',
            'body' => 'required',
        ]);
        Blog::updateOrCreate(
            ['id' => $this->blog_id], [
                'title' => $this->title,
                'body' => $this->body
            ]);
        session()->flash('message',
            $this->blog_id ?
             'Blog mis à jour avec succès.'
              : 'Blog créé avec succès.');
        $this->closeModal();
        $this->resetInputFields();
    }
    public function edit($id)
    {
        $blog = Blog::findOrFail($id);
        $this->blog_id = $id;
        $this->title = $blog->title;
        $this->body = $blog->body;
        $this->openModal();
    }
    public function delete($id)
    {
        Blog::find($id)->delete();
        session()->flash('message',
        'Blog supprimé avec succès.');
    }
}
```

- **Mettre à jour les fichiers Blade**

1. **Fichier `blogs.blade.php`**

- Ici, nous mettrons à jour la liste suivante de fichiers pour notre page de liste, créer une page.
- Alors, mettons à jour le fichier suivant: **resources/views/livewire/blogs.blade.php**

apcpedagogie.com

```blade
<x-slot name="header">
    <h2 class="font-semibold text-xl text-gray-800 leading-tight">
        livewire jetstream projet01 -apcpedagogie.com)
    </h2>
</x-slot>
<div class="py-12">
    <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
        <div class="bg-white overflow-hidden shadow-xl sm:rounded-lg px-4 py-4">
            @if (session()->has('message')) .teal-500 rounded-b text-teal-900
                <div class="bg-teal-100 border-t-4 border-
                px-4 py-3 shadow-md my-3" role="alert">
                  <div class="flex">
                    <div>
                      <p class="text-sm">{{ session('message') }}</p>
                    </div>
                  </div>
                </div>
            @endif                          hover:bg-blue-700 text-white
            <button wire:click="create()" class="bg-blue-500
            font-bold py-2 px-4 rounded my-3">Créer un nouveau blog</button>
            @if($isOpen)
                @include('livewire.create')
            @endif
            <table class="table-fixed w-full">
                <thead>
                    <tr class="bg-gray-100">
                        <th class="px-4 py-2 w-20">No.</th>
                        <th class="px-4 py-2">Title</th>
                        <th class="px-4 py-2">Body</th>
                        <th class="px-4 py-2">Action</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach($blogs as $blog)
                    <tr>
                        <td class="border px-4 py-2">{{ $blog->id }}</td>
                        <td class="border px-4 py-2">{{ $blog->title }}</td>
                        <td class="border px-4 py-2">{{ $blog->body }}</td>
                        <td class="border px-4 py-2">
                        <button wire:click="edit({{ $blog->id }})"
                            class="bg-blue-500 hover:bg-blue-700 text-white
                            font-bold py-2 px-4 rounded">Modifier</button>
                           <button wire:click="delete({{ $blog->id }})"
                            class="bg-red-500 hover:bg-red-700 text-white
                            font-bold py-2 px-4 rounded">Supprimer</button>
                        </td>
                    </tr>
                    @endforeach
                </tbody>
            </table>
        </div>
    </div>
</div>
```

- **Fichier `create.blade.php`**

```
<div class="fixed z-10 inset-0 overflow-y-auto ease-out duration-400">
    <div class="flex items-end justify-center min-h-screen pt-4 px-4 pb-20 text-center sm:block
sm:p-0">
        <div class="fixed inset-0 transition-opacity">
            <div class="absolute inset-0 bg-gray-500 opacity-75"></div>
        </div>
        <!-- Cet élément consiste à inciter le navigateur à centrer le contenu modal. -->
        <span class="hidden sm:inline-block sm:align-middle sm:h-screen"></span>?
        <div class="inline-block align-bottom bg-white rounded-lg text-left overflow-hidden
shadow-xl transform transition-all sm:my-8 sm:align-middle sm:max-w-lg sm:w-full" role="dialog"
aria-modal="true" aria-labelledby="modal-headline">
            <form>
                <div class="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4">
                    <div class="">
                        <div class="mb-4">
                            <label for="exampleFormControlInput1" class="block text-gray-700
text-sm font-bold mb-2">Title:</label>
                            <input type="text" class="shadow appearance-none border rounded w-
full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"
id="exampleFormControlInput1" placeholder="Enter Title" wire:model="title">
                            @error('title') <span class="text-red-500">{{ $message
}}</span>@enderror
                        </div>
                        <div class="mb-4">
                            <label for="exampleFormControlInput2" class="block text-gray-700
text-sm font-bold mb-2">Body:</label>
                            <textarea class="shadow appearance-none border rounded w-full py-2
px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"
id="exampleFormControlInput2" wire:model="body" placeholder="Enter Body"></textarea>
                            @error('body') <span class="text-red-500">{{ $message
}}</span>@enderror
                        </div>
                    </div>
                </div>
                <div class="bg-gray-50 px-4 py-3 sm:px-6 sm:flex sm:flex-row-reverse">
                    <span class="flex w-full rounded-md shadow-sm sm:ml-3 sm:w-auto">
                        <button wire:click.prevent="store()" type="button" class="inline-flex
justify-center w-full rounded-md border border-transparent px-4 py-2 bg-green-600 text-base
leading-6 font-medium text-white shadow-sm hover:bg-green-500 focus:outline-none focus:border-
green-700 focus:shadow-outline-green transition ease-in-out duration-150 sm:text-sm sm:leading-
5">
                        Save
                        </button>
                    </span>
```

apcpedagogie.com

```
                    <span class="mt-3 flex w-full rounded-md shadow-sm sm:mt-0 sm:w-auto">
                        <button wire:click="closeModal()" type="button" class="inline-flex
justify-center w-full rounded-md border border-gray-300 px-4 py-2 bg-white text-base leading-6
font-medium text-gray-700 shadow-sm hover:text-gray-500 focus:outline-none focus:border-blue-300
focus:shadow-outline-blue transition ease-in-out duration-150 sm:text-sm sm:leading-5">
                            Cancel
                        </button>
                    </span>
                </div>
            </form>
        </div>
    </div>
</div>
```

- **Mettre à jour les routes "`routes/web.php`"**

```
use App\Http\Livewire\Blogs;
Route::get('blog', Blogs::class);
```

- **Exécuter le projet**
  - Vous pouvez maintenant ouvrir l'URL ci-dessous sur votre navigateur:
    - `php artisan serve`
    - `localhost:8000/blog`
  - **Le dashboard**
    - On a un menu déroulant à droite en cliquant sur l’image (par défaut on a les deux premiers caractères du nom mais on va pouvoir mettre une vraie image) :
    - Là on peut procéder à 3 actions :

1. accéder au profil
2. accéder aux tokens des API
3. se déconnecter

- **Profil**
  - Les photos de profil de Laravel 8 Jetstream ne s'affichent pas

apcpedagogie.com

1. Localisez le fichier dans le dossier de configuration appelé **jetstream.php** et recherchez la clé

```
43
44      'features' => [
45          // Features::termsAndPrivacyPolicy(),
46          Features::profilePhotos(),
47          // Features::api(),
48          // Features::teams(['invitations' => true]),
49          Features::accountDeletion(),
50      ],
51
```

ofilePhotos()

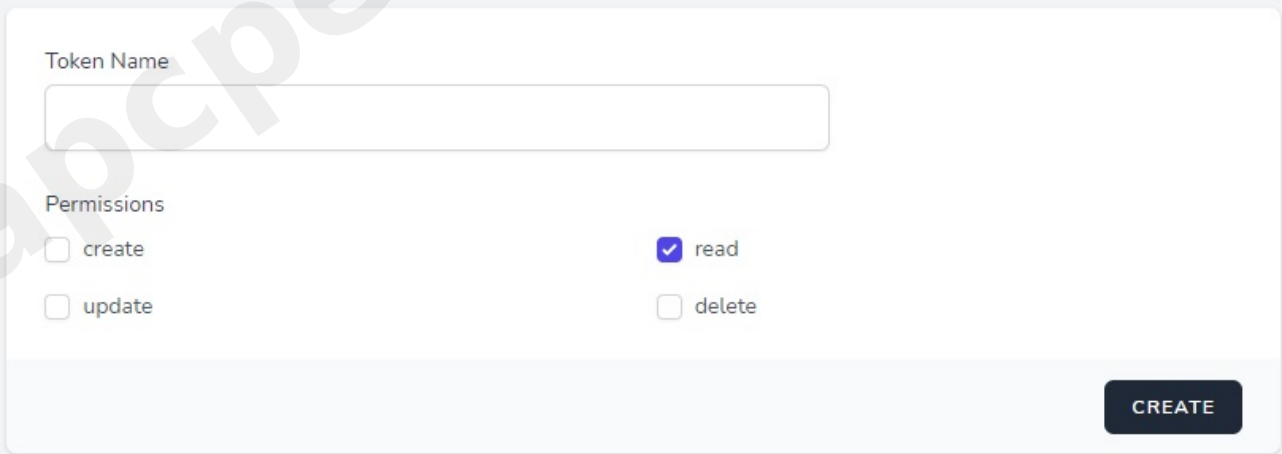
Accédez à **.env** et

remplacez l'uri de l'application par cette **APP_URL=http://localhost:8000**

2. Dans le terminal entrez : `php artisan serve`

- **Les API**

  - Jetstream est équipé pour l'utilisation de Laravel Sanctum, qui est un package pour Laravel qui permet de mettre en place simplement des API à partir de l'utilisation de tokens.
  - Pour l'activer
  - Localisez le fichier dans le dossier de configuration appelé **jetstream.php** et recherchez la clé '**features**' dans le tableau renvoyé et décommentez les `Features::api()`
  - Un utilisateur peut générer des tokens qui permettent un accès à une API avec des permissions bien définies. Sanctum est installé en même temps que Jetstream et est donc directement disponible.
  - Jetstream propose une interface simple pour la génération des tokens et de leurs permissions :



Sommaire du cours Laravel

Source : https://apcpedagogie.com/laravel-8-jetstream-livewire-crud · téléchargé le 03/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com