Listes déroulantes dépendantes avec Symfony 8 et Livewire

Le Concept : L'Effet Papillon Graphique (Reactive Cascade)

La mise en place de listes déroulantes dépendantes (par exemple, sélectionner un Pays qui filtre dynamiquement la liste des Villes) requiert traditionnellement l'écriture de scripts de routage d'API (`fetch` ou `Axios`) couplés à des écouteurs d'événements JavaScript complexes.

Avec **Symfony UX LiveComponent**, nous basculons sur un paradigme réactif unifié. Le serveur conserve l'autorité absolue sur les règles métiers et l'état de la base de données. Le changement de la valeur "maître" déclenche une mutation d'état asynchrone transparente qui recalcule, filtre et ré-émet le fragment HTML de la liste "esclave" via une mise à jour ciblée du DOM.

Le Composant PHP : Isolation d'État et Hooks de Cycle de Vie

Pour parer à toute incohérence de persistance, nous exploitons la convention de hook de mise à jour de Symfony UX : `updatedPropertyName()`. Si l'utilisateur change de pays, la ville précédemment sélectionnée doit être immédiatement réinitialisée à `null` pour éviter l'envoi d'un couple Pays/Ville incohérent au serveur.

```
<?php

namespace App\Twig\Components;

use App\Repository\CountryRepository;
use App\Repository\CityRepository;
use Symfony\UX\LiveComponent\Attribute\AsLiveComponent;
use Symfony\UX\LiveComponent\Attribute\LiveProp;
use Symfony\UX\LiveComponent\DefaultActionTrait;

#[AsLiveComponent('dependent_dropdown')]
class DependentDropdown
{
    use DefaultActionTrait;

    // Propriété maître synchronisée bidirectionnellement
    #[LiveProp(writable: true)]
```

apcpedagogie.com

```
    public ?int $countryId = null;

    // Propriété esclave réinitialisée lors des mutations du parent
    #[LiveProp(writable: true)]
    public ?int $cityId = null;

    public function __construct(
        private CountryRepository $countryRepository,
        private CityRepository $cityRepository
    ) {}

    /**
     * Hook de cycle de vie de LiveComponent.
     * Déclenché automatiquement immédiatement APRES la modification de $countryId.
     */
    public function updatedCountryId(): void
    {
        // Mesure de sécurité : Remise à zéro de l'élément esclave pour éviter les
désynchronisations
        $this->cityId = null;
    }

    /**
     * Retourne la liste globale des pays pour alimenter le premier select.
     */
    public function getCountries(): array
    {
        return $this->countryRepository->findBy([], ['name' => 'ASC']);
    }

    /**
     * Calcule dynamiquement la liste des villes filtrées.
     * Cette méthode est ré-évaluée à chaque cycle de rafraîchissement du composant.
     */
    public function getCities(): array
    {
        if (!$this->countryId) {
            return [];
        }

        return $this->cityRepository->findBy(
            ['country' => $this->countryId],
            ['name' => 'ASC']
```

apcpedagogie.com

```
        );
    }
}
```



ıdication Graphique

Nous encapsulons les balises `select` dans une architecture Tailwind épurée. L'utilisation intelligente des modificateurs de chargement `data-loading="attr(disabled)"` permet de figer l'interface utilisateur pendant le traitement réseau de la requête en arrière-plan.

```
<!-- templates/components/dependent_dropdown.html.twig -->
<div {{ attributes }} class="w-full max-w-xl bg-white border border-slate-200 rounded-2xl
shadow-sm p-6">

    <div class="mb-5 flex justify-between items-center">
        <h3 class="text-base font-bold text-slate-800 m-0">Localisation Géographique</h3>
        <!-- Indicateur asynchrone masqué par défaut -->
        <div data-loading class="text-xs text-indigo-600 font-medium flex items-center gap-1.5
animate-pulse">
            Chargement des zones...
        </div>
    </div>

    <div class="space-y-4">
        <!-- Sélecteur 1 : Le Pays (Composant Maître) -->
        <div class="flex flex-col gap-1.5">
```

```
            <label for="country-select" class="text-xs font-bold text-slate-500 uppercase
tracking-wider">Pays de résidence</label>
            <select
                id="country-select"
                data-model="countryId"
                class="w-full bg-slate-50 border border-slate-200 rounded-xl p-3 text-sm text-
slate-700 focus:bg-white focus:ring-2 focus:ring-indigo-500/20 focus:border-indigo-500
transition-all outline-none"
            >
                <option value="">-- Choisir un pays d'ancrage --</option>
                {% for country in this.countries %}
                    <option value="{{ country.id }}">{{ country.name }}</option>
                {% endfor %}
            </select>
        </div>

        <!-- Sélecteur 2 : La Ville (Composant Esclave Dépendant) -->
        <div class="flex flex-col gap-1.5">
            <label for="city-select" class="text-xs font-bold text-slate-500 uppercase tracking-
wider">Ville de rattachement</label>
            <select
                id="city-select"
                data-model="cityId"
                data-loading="attr(disabled)"
                {% if not countryId %}disabled{% endif %}
                class="w-full bg-slate-50 border border-slate-200 rounded-xl p-3 text-sm text-
slate-700 focus:bg-white focus:ring-2 focus:ring-indigo-500/20 focus:border-indigo-500
transition-all outline-none disabled:opacity-50 disabled:cursor-not-allowed"
            >
                {% if not countryId %}
                    <option value="">Veuillez d'abord sélectionner un pays...</option>
                {% else %}
                    <option value="">-- Choisir une ville --</option>
                    {% for city in this.cities %}
                        <option value="{{ city.id }}">{{ city.name }}</option>
                    {% endfor %}
                {% endif %}
            </select>
        </div>
    </div>
</div>
```

apcpedagogie.com

Analyse des Avantages Architecturaux

Pourquoi cette implémentation surclasse les approches traditionnelles :

- **Zéro API Boilerplate :** Vous n'avez pas besoin d'écrire un contrôleur d'API dédié (ex: `CityApiController`), de sérialiser vos entités en JSON, ou de gérer les codes de statut HTTP. L'ensemble des transferts de données s'effectue dans l'écosystème natif de Symfony.
- **Immunité contre la corruption d'état :** L'utilisation du hook de cycle de vie `updatedCountryId()` élimine l'effet de bord classique des architectures asynchrones où le sélecteur esclave conserve une valeur incohérente liée à un ancien parent obsolète.
- **Verrouillage UI natif (UX Guard) :** L'instruction `data-loading="attr(disabled)"` garantit une intégrité transactionnelle parfaite : l'utilisateur ne peut pas soumettre ou altérer la liste déroulante esclave tant que le serveur n'a pas finalisé le calcul géométrique de filtrage des données.

Résumé du Module

La gestion des listes en cascade réactives via **Symfony UX LiveComponent** met en lumière la puissance des architectures server-driven modernes. En déportant la gestion d'état et les hooks directement au sein de votre classe PHP, vous simplifiez considérablement votre base de code tout en garantissant un niveau de performance et de fluidité identique à celui d'une application développée sous Vue.js ou React.

*Rédigé et structuré par : **Med Khalil Kribi***

Source : https://apcpedagogie.com/listes-deroulantes-dependantes-avec-symfony-8-et-livewire · téléchargé le 03/11/2026 · © APC Pédagogie · apcpedagogie.com