

ADRAR\o/ **PÔLE NUMÉRIQUE**

- > **INFRASTRUCTURES** SYSTÈMES & RÉSEAUX
- > **CYBERSÉCURITÉ** INFRASTRUCTURES & APPLICATIONS
- > **DEVOPS / SCRIPTING** & AUTOMATISATION
- > **DEVELOPPEMENT** WEB & MOBILE
- > **TRANSFORMATION NUMERIQUE DES ENTREPRISES**

www.adrar-numerique.com

Le concept Symfony

L'UX initiative est un ensemble de bibliothèques permettant d'intégrer de manière transparente des outils JS.

Nous allons voir l'exemple avec trois bibliothèques UX:

- UX Chart.js: permet de générer des graphs
- UX Autocomplete.js: permet par exemple d'afficher une liste de résultats pour un texte donné
- UX LazyLoader.js: permet de charger les photos petit à petit pour alléger le poids des requêtes

NB: Le JS est automatiquement géré pour vous, il ne faut traiter que le back.

Installation Symfony

Pour utiliser ces bibliothèques, nous allons devoir installer plusieurs éléments.

Dans un premier temps, vérifiez la présence de node:

```
$ npm --version
```

 présent,

installez-le.

Ensuite, installez le WebPack Encore:

```
$ composer require symfony/webpack-encore-bundle
```

```
$ npm install
```

→ ceci va créer un dossier "assets", "node_modules" et un fichier "webpack.config.js" à la racine du projet

A présent, nous sommes en mesure de télécharger des bibliothèques UX.

ChartJS

Tapez cette commande pour ajouter la dépendance au projet:

```
$ composer require symfony/ux-chartjs
```

Pour compresser l'ensemble des fichiers css et js:

```
$ npm install && npm run build
```

Rendez-vous dans le fichier .env pour commenter la ligne **DATABASE_URL** et dé-commenter celle avec le sqlite (driver à activer dans le php.ini) puis exécuter la commande pour créer la BDD

```
$ composer console doctrine:database:create
```

Créer une entité qui va porter deux attributs: une date (type date) et une valeur (type integer), effectuer la migration et insérer des enregistrements (make:crud...) Une fois cela fait, créer une nouvelle route qui va récupérer l'ensemble des données via le *repository* (code ci-après à adapter)

ChartJS

```
#[Route('/chartjs', name: 'app_chartjs')]
public function index(DailyResultRepository $dailyResultRepository, ChartBuilderInterface $chartBuilder): Response
{
    $dailyResults = $dailyResultRepository->findAll();
    $labels = [];
    $data = [];

    foreach ($dailyResults as $dailyResult) {
        $labels[] = $dailyResult->getDate()->format('d/m/Y');
        $data[] = $dailyResult->getValue();
    }

    $chart = $chartBuilder->createChart(Chart::TYPE_LINE);
    $chart->setData([
        'labels' => $labels,
        'datasets' => [
            [
                'label' => 'My first dataset',
                'backgroundColor' => 'rgb(255, 99, 132)',
                'borderColor' => 'rgb(255, 99, 132)',
                'data' => $data
            ]
        ]
    ]);
    $chart->setOptions([/* */]);

    return $this->render('chartjs/index.html.twig', [
        'chart' => $chart,
    ]);
}
```

ChartJsController.php

ChartJS

Rendez-vous sur l'url créée et contemplez le joli graphique qui se dessine devant vos yeux ébahis



UX Autocomplete Symfony

AUTOCOMPLETE

Tapez cette commande pour ajouter la dépendance au projet:

```
$ composer require symfony/ux-autocomplete
```

Pour compresser l'ensemble des fichiers css et js:

```
$ npm install && npm run build
```

Nous allons voir deux cas d'usage: sans et avec AJAX

AUTOCOMPLETE (sans ajax)

Vous remarquerez que dans le formulaire, nous ne mettons qu'un paramètre "autocomplete", automatiquement, cela fonctionnera:

```

public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add('date', DateTimeType::class)
        ->add('value', IntegerType::class)
        ->add('choices', ChoiceType::class, [
            'choices' => [
                'Choose a portion size' => '',
                'small' => 's',
                'medium' => 'm',
                'large' => 'l',
                'extra large' => 'xl',
                'all you can eat' => '∞',
            ],
            'autocomplete' => true,
            'mapped' => false
        ]);
}
    
```

DailyResultACType.php

| | |
|--------------------|-------------|
| Choisir une taille | larg |
| small | large |
| medium | extra large |
| large | |
| extra large | |
| Tout ce que tu as | |

AUTOCOMPLETE (avec ajax)

Pour générer le formulaire, utilisez la commande:

```
$ symfony console make:autocomplete-field
```

```

public function configureOptions(OptionsResolver $resolver)
{
    $resolver->setDefaults([
        'class' => DailyResult::class,
        'placeholder' => 'Choose a DailyResult',
        'choice_label' => 'value',
        'query_builder' => function(DailyResultRepository $dailyResultRepository) {
            return $dailyResultRepository->createQueryBuilder('dailyResult');
        },
        // 'security' => 'ROLE_SOMETHING',
    ]);
}
    
```

DailyResultACType.php

```

{% block body %}
    {{ form(form) }}
{% endblock %}
    
```

autocomplete/index.html.twig

Choose a DailyResult

45

4554

44

7

No more results

UX LazyLoader Symfony

Tapez cette commande pour ajouter la dépendance au projet:

```
$ composer require symfony/ux-lazy-image
```

Pour compresser l'ensemble des fichiers css et js:

```
$ npm install && npm run build
```

LazyLoader

```

```

LazyLoader

Tapez cette commande pour ajouter la dépendance au projet:

```
$ composer require intervention/image
```

```
$ composer require kornrunner/blurhash
```

Il faut augmenter la valeur de “memory_limit” dans votre php.ini (256 au lieu de 128 pour tester).

Vous devriez remarquer que le chargement est extrêmement plus lent de la sorte...

```

```