# **Symfony 6** : introduction

**Achref El Mouelhi**

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

# Plan

# Symfony

## Framework

- Plusieurs traductions
  - cadriciel
  - environnement de développement
  - cadre d'applications
  - ...
- Ensemble de composants logiciels
- Facilitant la réalisation d'une (partie de l') application
- Imposant une certaine structure, logique, syntaxe...

# Symfony

## Plusieurs types de Framework

- Frameworks applicatifs pour le développement d'applications web :
  - **Symfony** pour **PHP**
  - **Angular** pour **JavaScript**,
  - **Spring** pour **Java**,
  - ...
- Frameworks de présentation de contenu web :
  - **Bootstrap** pour **CSS**
- Frameworks de persistance (**ORM**) comme **Doctrine**, **Hibernate**...
- Frameworks de logging comme **Log4j**
- Frameworks de test comme **phpUnit**
- ...

# Symfony

## Framework : avantages

- Gain de temps
- Code de qualité
- Meilleure organisation de projet
- Respect de bonnes pratiques et utilisation de design patterns
- Faciliter le travail d'équipe
- Conflits entre dépendances gérés par le framework
- Composants et API à disposition des développeurs
- ...

# Symfony

## Frameworks : inconvénients

- Complexité
- Apprentissage
- ...

# Symfony

## Symfony

- framework **PHP** sorti en octobre 2005
- français
- conçu et développé par **SensioLabs**
- open-source
- basé sur l'architecture **MVC**
- utilisant le protocole **HTTP**

# Symfony

## Symfony

- Intégrant plusieurs composants
  - **HttpFoundation** : contenant les classes relatives au protocole **HTTP** comme **Request** et **Response**
  - **Console** : facilitant la génération de plusieurs autres composants (formulaires, entités...)
  - **Form** : facilitant la construction, la récupération et le contrôle de données saisies
  - Liste complète (`https://symfony.com/components`)
- Nécessitant une bonne connaissance de
  - **HTTP**
  - **POO**
  - **ORM**
  - ...

# Symfony

## Exemple d'utilisation de **Symfony**

- Dailymotion (depuis 2009)
- Composer
- OpenClassroom
- SNCF
- ...

# Symfony

## Les différentes versions de **Symfony**

- **Symfony 1** : sorti en octobre 2005
- **Symfony 2** : sorti en août 2011
- **Symfony 3** : sorti en novembre 2015
- **Symfony 4** : sorti en novembre 2017
- **Symfony 5** : sorti en novembre 2019
- **Symfony 6** : sorti en novembre 2021

# Symfony

## Utiliser un Environnement de développement intégré (**IDE**) ?

- Console auto-intégrée
- Auto-complétion
- Auto-compilation
- Coloration syntaxique
- Meilleure structuration du projet
- Meilleure qualité du code

# Symfony

## Quelques **IDE** pour **PHP**

- **Visual Studio Code**
- Eclipse
- PHP Storm
- ...

# Symfony

## **Visual Studio Code** (ou **VSC**), pourquoi ?

- Gratuit
- Pouvant s'adapter selon le langage de programmation
- Extensible via l'installation de quelques centaines d'extensions

© Achref El

# Symfony

## **Visual Studio Code** (ou **VSC**), pourquoi ?

- Gratuit
- Pouvant s'adapter selon le langage de programmation
- Extensible via l'installation de quelques centaines d'extensions

© Achref E

## **VSC** : téléchargement

`code.visualstudio.com/download`

# Symfony

## Quelques recommandations (raccourcis) pour **VSC**

- Pour activer la sauvegarde automatique : aller dans `File > AutoSave`
- Pour indenter son code : `Alt` `Shift` `f`
- Pour commenter/décommenter : `Alt` `Shift` `:`
- Pour faire une sélection multiple : `Ctrl` `f2`
- Pour dupliquer une sélection : `Ctrl` `d`

# Symfony

## Extension **VSC** pour **PHP**

**PHP Intelephense** pour l'auto-complétion et le formatage du code.

© Achref EL MOUEL

# Symfony

## Extension **VSC** pour **PHP**

**PHP Intelephense** pour l'auto-complétion et le formatage du code.

Prof EL MOUE

## Extension **VSC** pour **Symfony**

**PHP Namespace Resolver** pour automatiser l'import des namespaces.

# Symfony

## **WAMP**, pourquoi ?

**WAMP** intègre :

- Un serveur web (**Apache**)
- Un interpréteur du langage **PHP**
- Et probablement un système de gestion de base de données (généralement **MySQL**)

# Symfony

## Avant d'installer **WAMP**

Il faut aller à `https://wampserver.aviatechno.net/`, télécharger **All VC Redistribuable Packages (x86_x64) (32 & 64bits) MD5** puis décompresser et tout installer.

© Achref EL M

# Symfony

## Avant d'installer **WAMP**

Il faut aller à `https://wampserver.aviatechno.net/`, télécharger **All VC Redistribuable Packages (x86_x64) (32 & 64bits) MD5** puis décompresser et tout installer.

## **WAMP** : installation

`http://www.wampserver.com/`

# Symfony

## Première utilisation de **WAMP**

- Démarrer **WAMP**
- Cliquer sur **WAMP** dans la barre de démarrage et choisir `Redémarrer les services`
- Si l'icône de **WAMP** n'est pas en vert, aller vérifier `http://forum.wampserver.com/read.php?1,88043`

# Symfony

## Pour intégrer **PHP 8** avec **WAMP 3.2.3**

- Aller à `https://wampserver.aviatechno.net/` et télécharger puis installer **Wampserver update 3.2.5 MD5**
- Aller à `https://wampserver.aviatechno.net/` et télécharger puis installer **PHP 8.0.8 64 bit x64 MD5**
- Redémarrer les services de **WAMP**

# Symfony

## Quelques éléments dans le menu de démarrage de **WAMP**

- `localhost` : page de démarrage de **WAMP**
- `phpMyAdmin` : page web permettant la gestion des bases de données **MySQL**
- Répertoire `www` : emplacement des projets **PHP** sur le disque dur
- ...

# Symfony

## Avant toute installation

- Pour **Symfony 5.3**, il faut une version de **PHP** $\geq$ 7.2.5
- Pour **Symfony 6.0** (version stable), il faut une version de **PHP** $\geq$ 8.0.2
- Pour **Symfony 6.1** (en développement), il faut une version de **PHP** $\geq$ 8.1.0

© Achref EL

# Symfony

## Avant toute installation

- Pour **Symfony 5.3**, il faut une version de **PHP** $\geq$ 7.2.5
- Pour **Symfony 6.0** (version stable), il faut une version de **PHP** $\geq$ 8.0.2
- Pour **Symfony 6.1** (en développement), il faut une version de **PHP** $\geq$ 8.1.0

© Achref EL

## Ce n'est pas terminé

Il faut aussi **Composer** pour l'installation de packages.

# Symfony

## Composer ?

- Gestionnaire de dépendance de **PHP**
- écrit en **PHP**

© Achref EL MOU

# Symfony

## Composer ?

- Gestionnaire de dépendance de **PHP**
- écrit en **PHP**

## Installation de **Composer**

- Allez à `https://getcomposer.org/Composer-Setup.exe`
- Lancez l'installation (de **Composer**) depuis le fichier téléchargé

# Symfony

## Installation sous **Windows** (avec un installer **Symfony**)

Téléchargez `https://get.symfony.com/cli/setup.exe` puis installez

© Achref EL MOUELHI ©

# Symfony

## Installation sous **Windows** (avec un installer **Symfony**)

Téléchargez `https://get.symfony.com/cli/setup.exe` puis installez

## Installation sous **Linux** (avec une commande)

```
wget https://get.symfony.com/cli/installer -O - | bash
```

© AU... QUELHI ©

# Symfony

## Installation sous **Windows** (avec un installer **Symfony**)

Téléchargez `https://get.symfony.com/cli/setup.exe` puis installez

## Installation sous **Linux** (avec une commande)

```
wget https://get.symfony.com/cli/installer -O - | bash
```

## Installation sous **Mac** (avec une commande)

```
curl -sS https://get.symfony.com/cli/installer | bash
```

© AU... SOUELHI ©

# Symfony

## Création d'un projet **Symfony** 6 : deux solutions

- Avec **Symfony CLI** en utilisant la commande `symfony ...`
- Avec **Composer** en utilisant la commande `composer ...`

# Symfony

**Première solution : créer un projet complet Symfony 6 en utilisant Symfony CLI**

```
symfony new --full cours_symfony
```

© Achref EL MOUELHI ©

# Symfony

**Première solution : créer un projet complet Symfony 6 en utilisant Symfony CLI**

```
symfony new --full cours_symfony
```

**Première solution : créer un projet Web Symfony 6 (avec le minimum de paquets) en utilisant Symfony CLI**

```
symfony new --webapp cours_symfony
```

© AU... OUELHI ©

# Symfony

**Première solution : créer un projet complet Symfony 6 en utilisant Symfony CLI**

```
symfony new --full cours_symfony
```

**Première solution : créer un projet Web Symfony 6 (avec le minimum de paquets) en utilisant Symfony CLI**

```
symfony new --webapp cours_symfony
```

**Première solution : créer un projet console ou microservice Symfony 6**

```
symfony new cours_symfony
```

# Symfony

**Pour créer un projet Web Symfony d'une version antérieure**

```
symfony new --full cours_symfony --version=4.4
```

© Achref EL MOUEL

# Symfony

**Pour créer un projet Web Symfony d'une version antérieure**

```
symfony new --full cours_symfony --version=4.4
```

**Pour créer un projet Web Symfony et utiliser la dernière version stable**

```
symfony new --full cours_symfony --version=lts
```

# Symfony

**Deuxième solution : créer un projet Web Symfony 6 en utilisant Composer**

```
composer create-project symfony/website-skeleton cours_symfony
```

© Achref EL MOUELHI ©

# Symfony

**Deuxième solution : créer un projet Web Symfony 6 en utilisant Composer**

```
composer create-project symfony/website-skeleton cours_symfony
```

**Pour créer un projet console ou microservice Symfony 6 en utilisant Composer**

```
composer create-project symfony/skeleton cours_symfony
```

© Achr QUELHI ©

# Symfony

**Deuxième solution : créer un projet Web Symfony 6 en utilisant Composer**

```
composer create-project symfony/website-skeleton cours_symfony
```

**Pour créer un projet console ou microservice Symfony 6 en utilisant Composer**

```
composer create-project symfony/skeleton cours_symfony
```

© Achr QUELHI ©

**Pour créer un projet Web Symfony d'une version antérieure en utilisant Composer**

```
composer create-project symfony/website-skeleton:"^4.4" cours_symfony
```

# Symfony

## Pour vérifier la version utilisée de **Symfony**, exécutez

- `php app/console --version` pour **Symfony** 1 et 2
- `php bin/console --version` depuis la version 3

© Achref EL MOUELHI

# Symfony

## Pour vérifier la version utilisée de **Symfony**, exécutez

- `php app/console --version` pour **Symfony** 1 et 2
- `php bin/console --version` depuis la version 3

**Pour connaitre la version de PHP utilisée par Symfony, exécutez**

```
symfony php -v
```

# Symfony

## Pour vérifier la version utilisée de **Symfony**, exécutez

- `php app/console --version` pour **Symfony** 1 et 2
- `php bin/console --version` depuis la version 3

**Pour connaitre la version de PHP utilisée par Symfony, exécutez**

```
symfony php -v
```

**Pour connaitre la version de Symfony CLI, exécutez**

```
symfony -V
```

# Symfony

**Pour connaitre tous les détails sur Symfony, PHP, l'necodage..., exécutez**

```
php bin/console about
```

© Achref EL MOU

# Symfony

**Pour connaitre tous les détails sur Symfony, PHP, l'necodage..., exécutez**

```
php bin/console about
```

**Pour connaitre la version des modules Composer d'un projet Symfony, exécutez**

```
composer show
```

# Symfony

## Pour lancer un projet **Symfony** 6 créé avec la commande `symfony`

- Allez dans le répertoire du projet (`cd cours_symfony`)
- Exécutez `symfony server:start` ou `symfony serve`

# Symfony

## Pour lancer un projet **Symfony** 6 créé avec la commande `symfony`

- Allez dans le répertoire du projet (`cd cours_symfony`)
- Exécutez `symfony server:start` ou `symfony serve`

**Pour lancer un projet Symfony en arrière-plan**

```
symfony serve -d
```

# Symfony

## Pour lancer un projet **Symfony** 6 créé avec la commande `symfony`

- Allez dans le répertoire du projet (`cd cours_symfony`)
- Exécutez `symfony server:start` ou `symfony serve`

**Pour lancer un projet Symfony en arrière-plan**

```
symfony serve -d
```

**Pour afficher les données du log**

```
symfony server:log
```

# Symfony

**Pour utiliser un projet Symfony hébergé sur GitHub**

```
# cloner un projet existant
git clone lien_vers_repository_github

# se positionner dans le projet
cd my-project/

# installer les dépendances composer dans vendor
composer install
```

# Symfony

## Sous **Visual Studio Code**

- Installer l'extension **PHP Namespace Resolver**
- Elle permet d'importer les namespaces nécessaires pour certaines classes
- Pour l'utiliser, faites clic droit sur la classe concernée
- Allez dans `Import Class` et choisir le namespace correspondant

# Symfony

## Structure d'un projet **Symfony 6**

- `assets/` : pour les fichiers statiques (**CSS**...) [projet webapp]
- `bin/` : contenant deux exécutables, la **console** de **Symfony** et **PHPUnit**
- `config/` : pour les fichiers de configuration (routes, **ORM**...)
- `migrations/` : pour enregistrer les données de migration de la base de données
- `public/` (appelé `web` dans la version 3) : seul dossier accessible de l'extérieur (contenant le contrôleur frontal `index.php`)
- `src/` : pour les fichiers sources de type classes (contrôleurs, entités, formulaires, **DAO**...)
- `templates/` : contenant les vues (vue partielle) de l'application
- `tests/` : pour les fichiers de test [projet full]
- `translations/` : pour les fichiers de l'internationalisation
- `var/` (utilisé pendant l'exécution) : contenant les données de cache, le log, les sessions...
- `vendor/` : contenant les packages nécessaires pour **Symfony** (mentionnés dans `composer.json`)

# Symfony

## Kernel ?

- noyau de **Symfony**
- défini dans `src/Kernel.php` (`app/AppKernel.php` dans **Symfony 3**)
- utilisé par le contrôleur frontal pour désigner le contrôleur adéquat pour répondre à la requête **HTTP** reçue

# Symfony

## Kernel ?

- noyau de **Symfony**
- défini dans `src/Kernel.php` (`app/AppKernel.php` dans **Symfony 3**)
- utilisé par le contrôleur frontal pour désigner le contrôleur adéquat pour répondre à la requête **HTTP** reçue

## Contrôleur frontal

- point d'entrée d'une application **Symfony**
- défini dans `public/index.php`

# Symfony

## Deux environnements de travail

- **prod** (destiné aux utilisateurs finaux de l'application)
  - montrant l'application telle qu'elle sera visible par les visiteurs
  - rapide à exécuter
  - n'affichant pas les messages d'erreur.
- **dev** (destinés aux développeurs)
  - Plus lent que la version de production
  - Environnement de débogage complet
  - Possibilité d'ajouter des nouvelles fonctionnalités

# Symfony

## Remarque

- Par défaut, une application est configurée à l'environnement **dev**
- Pour changer d'environnement, allez dans `.env` et mettez la valeur de `APP_ENV` à `prod`

# Symfony

**Extrait de** `public/index.php`

```
use App\Kernel;

require_once dirname(__DIR__).'/vendor/autoload_runtime.php';

return function (array $context) {
    return new Kernel($context['APP_ENV'], (bool) $context['APP_DEBUG']);
};
```

# Symfony

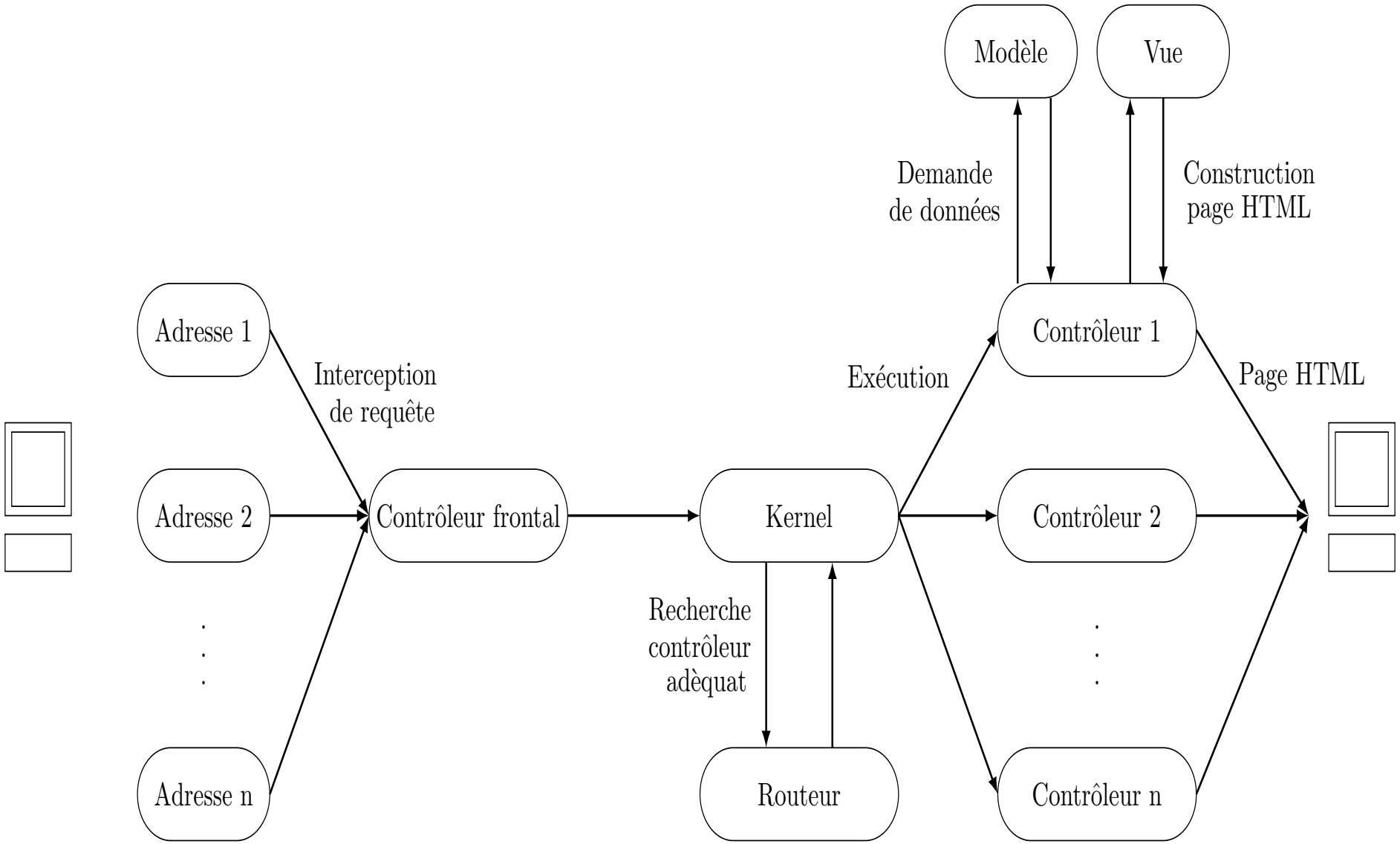
**Extrait de** `public/index.php`

```
use App\Kernel;

require_once dirname(__DIR__).'/vendor/autoload_runtime.php';

return function (array $context) {
    return new Kernel($context['APP_ENV'], (bool) $context['APP_DEBUG']);
};
```

## Explication

- Le contrôleur frontal charge le `kernel` selon l'environnement précisé dans `.env` (`APP_ENV=dev`).
- Le `kernel` retourne une réponse **HTTP** suite à la requête reçue.

# Symfony


Modèle
Vue
Demande de données
Construction page HTML
Adresse 1
Interception de requête
Exécution
Contrôleur 1
Page HTML
Adresse 2
Contrôleur frontal
Kernel
Contrôleur 2
Recherche contrôleur adèquat
Adresse n
Routeur
Contrôleur n

# Symfony

## Déroulement

- L'utilisateur saisit l'adresse d'une page de notre site (donc envoie une requête **HTTP**.
- Le contrôleur frontal intercepte la requête et il la transmet au **Kernel**.
- Ce dernier demande au Routeur le nom du contrôleur ainsi que l'action (méthode) associés à cette adresse.
- À la réception d'une réponse, le **Kernel** appelle le contrôleur.
- Le contrôleur communique avec le modèle pour récupérer ou stocker certaines données.
- Ensuite il renvoie ces données à la vue pour qu'elle construise la page **HTML** et la lui retourne.
- Enfin le contrôleur envoie une réponse **HTTP** contenant une page **HTML**.

# Symfony

## Quelques composants fondamentaux

- Console
- Flex

# Symfony

## Console

- Outil (composant) pour développeur
- Disponible depuis la première version de **Symfony**
- Permettant d'interagir avec l'application **Symfony** en ligne de commandes (**CLI** ou Command Line Interface).
- Défini dans le répertoire `bin` par le fichier `console`

# Symfony

## Console

- Outil (composant) pour développeur
- Disponible depuis la première version de **Symfony**
- Permettant d'interagir avec l'application **Symfony** en ligne de commandes (**CLI** ou Command Line Interface).
- Défini dans le répertoire `bin` par le fichier `console`

**Pour consulter la liste des commandes disponibles**

```
php bin/console
```

# Symfony

## La console, pourquoi ?

Pour :

- faciliter la création/génération de contrôleurs, entités, formulaires, bundles...
- éviter les erreurs
- accélérer le développement

# Symfony

## La console, pourquoi ?

Pour :

- faciliter la création/génération de contrôleurs, entités, formulaires, bundles...
- éviter les erreurs
- accélérer le développement

**Pour consulter la liste des éléments que l'on puisse générer avec la console**

```
php bin/console list make
```

# Symfony

## Flex

- Outil pour développeur
- Plugin **Composer** : écouteur d'évènement **Composer** (`install`, `update`, `remove`)
- Disponible depuis la version 4 de **Symfony**

# Symfony

## Flex

- Outil pour développeur
- Plugin **Composer** : écouteur d'évènement **Composer** (`install`, `update`, `remove`)
- Disponible depuis la version 4 de **Symfony**

## Rôle

- Configurer un package **Symfony** (ou Bundle) grâce à des configurateurs
- Faciliter la recherche d'un package grâce à des alias

# Symfony

## **Flex** utilise les **recipe** (recettes)

- contenant des informations sur les fichiers de configuration à créer ou modifier.
- contenant également une liste d'alias.

# Symfony

## **Flex** utilise les **recipe** (recettes)

- contenant des informations sur les fichiers de configuration à créer ou modifier.
- contenant également une liste d'alias.

## Deux dépôts **GitHub** de recettes

- dépôt officiel maintenu par l'équipe **Symfony** (marqué par **official**) : `https://github.com/symfony/recipes`
- dépôt de la communauté **Symfony** (marqué par **contrib**) : `https://github.com/symfony/recipes-contrib`
- Les deux sont regroupés dans `https://flex.symfony.com/` avec une zone de recherche.

# Symfony

**Alias de** `Api-platform` **:** (`https://github.com/symfony/recipes/blob/master/api-platform/api-pack/2.1/manifest.json`)

```
{
    "aliases": ["api", "api-platform"]
}
```

# Symfony

**Alias de** `Api-platform` **:** (`https://github.com/symfony/recipes/blob/master/api-platform/api-pack/2.1/manifest.json`)

```
{
    "aliases": ["api", "api-platform"]
}
```

**Pour l'installer**

```
composer require api-platform
```

# Symfony

**Alias de `Api-platform` : (`https://github.com/symfony/recipes/blob/master/api-platform/api-pack/2.1/manifest.json`)**

```
{
    "aliases": ["api", "api-platform"]
}
```

**Pour l'installer**

```
composer require api-platform
```

**Ou (grâce à Flex car aucun package Composer ne porte le nom `api`)**

```
composer require api
```

**Configurateur de** `monolog` **: (**`https://github.com/symfony/recipes/blob/master/api-platform/api-pack/2.1/manifest.json`**)**

```
{
    "bundles": {
        "Symfony\\Bundle\\MonologBundle\\MonologBundle": ["all"]
    },
    "copy-from-recipe": {
        "config/": "%CONFIG_DIR%/"
    },
    "aliases": ["log", "logger", "logging", "logs", "monolog"]
}
```

**Configurateur de `monolog` : (`https://github.com/symfony/recipes/blob/master/api-platform/api-pack/2.1/manifest.json`)**

```
{
    "bundles": {
        "Symfony\\Bundle\\MonologBundle\\MonologBundle": ["all"]
    },
    "copy-from-recipe": {
        "config/": "%CONFIG_DIR%/"
    },
    "aliases": ["log", "logger", "logging", "logs", "monolog"]
}
```

## Explication

- Type de configurateur : `copy-from-recipe`
- Effet : indique les fichiers à copier de la recette vers l'application **Symfony** (il faut copier le contenu du dossier `config/` provenant du recipe dans le dossier de configuration de l'application **Symfony**).
- Pour l'utiliser, il faut instancier `MonologBundle` pour tous les environements (`dev`, `prod`...)

# Symfony

## Autres configurateurs

- `copy-from-package` : indique les fichiers à copier du paquet **Composer** vers l'application **Symfony**
- `gitignore` : précise les entrées à ajouter à `.gitignore`
- `env` : indique les variables d'environnement à ajouter dans les fichiers `.env`
- `composer-scripts` : liste les commandes à ajouter dans la section scripts de `composer.json` pour qu'elles soient exécutées à la fin des commandes `install` et `update`
- ...