Module 04 - Symfony : Présentation générale et installation

Symfony 7.2 (2025)

Qu'est ce que Symfony ?

- Ensemble de composants open-source (+50 composants);
- Framework MVC mature et performant pour les applications web et CLI;

Démonstration: Installer le composant **var-dumper**

- Un **must-have** pour débug votre code (contexte web);
- Un `var_dump()` efficace et interactif (idéal pour le développement web)

Pourquoi utiliser un framework ?

- Structurer, standardiser le code source
- Travailler en équipe
- Éviter de réinventer la roue
- Produire un code plus facilement maintenable
- Augmenter la productivité

Quelques frameworks PHP

- Symfony, **open-source depuis 2005**, version actuelle: 6.1 (2022)
- Laminas project (ex Zend Framework)
- Laravel
- Slim
- CodeIgniter
- CakePHP
- MediaWiki
- API Platform, framework API PHP très avancé (toutes technologies confondues), intégré à Symfony
- etc.

CMS/Forums: Wordpress, Drupal Joomla!, Prestashop, MyBB, etc.

Portrait de Symfony

- **Support à long-terme**, un process de releases bien en place (rétro-comptabilité entre les versions mineurs, 3 ans de support sur toutes les versions majeures)
- **Philosophie**: améliorer la productivité des développeurs avec un environnement toujours à la pointe, rapide, flexible, basé sur des composants réutilisables, outils de debug et monitoring, incitation aux meilleures pratiques par design, etc.
- **Ressources**: grande communauté (+3000 core contributors, +600 000 développeur·ses), documentation riche, conférences/évènements, etc.
- **Interopérabilité**: respect des standards de la communauté PHP (PHPUnit, conventions de nommage, etc.) et basé sur des composants
- **Bonne réputation**: environnement stable et adopté par les professionnels
- **Désavantages**: assez complexe, demande de l'investissement
- > Utilisé par des grandes sociétés: Spotify, Blablacar, Vogue, Yahoo!, Dailymotion, etc.
- > Utilisé par d'autres framework PHP: Drupal (depuis la version 8), phpBB

Installation

- Installer Composer;
- Installer Symfony CLI un outil CLI pour gérer vos projets Symfony (créer des applications Symfony, développer sur un serveur web local, vérifier la sécurité de l'application, etc.). *Wrapper* autour de Composer
- Vérifier votre installation avec `symfony check:requirements`. Si des erreurs sont détectées (modules PHP manquants, etc.), corriger.

```
# Check que vous avez tous les pré-requis, suggestions de configuration de PHP
symfony check:requirements
# Check les versionde php installées et leur path sur la machine
symfony local:php:list
```

Création d'une web app

Créer un projet d'application web (template) avec Composer

```
symfony new app --version="7.2.*" --webapp
# Si pb de version, ne pas forcer la version et executer: symfony new app --webapp
cd app
# Checker votre version de Symfony
php bin/console --version
# Lancer un serveur web local hébérgeant l'application
symfony server:start -d
# Accéder au projet, ou se rendre à l'URL https://127.0.0.1:8000 (par défaut)
symfony open:local
```

Inspecter, obtenir de l'aide

Le programme `symfony` vous fournit énormément de commandes utiles. Apprenez à vous en servir.

```
# Aide sur une commande
symfony help [commande]
# Lister les serveurs en local
symfony server:ls
# Status du serveur
symfony server:status
```

Accéder à la base de données

Dans l'environnement de développement local, on peut *servir* la base de données via un conteneur Docker.

Installer Docker et Compose.

À la racine du projet

```
docker compose up -d
docker container ls #vérifier que le conteneur est lancé
# Se connecter à l'instance PostgreSQL en ligne de commande:  psql [dbname] [username]
symfony run psql app app
```

Commandes psql utiles:

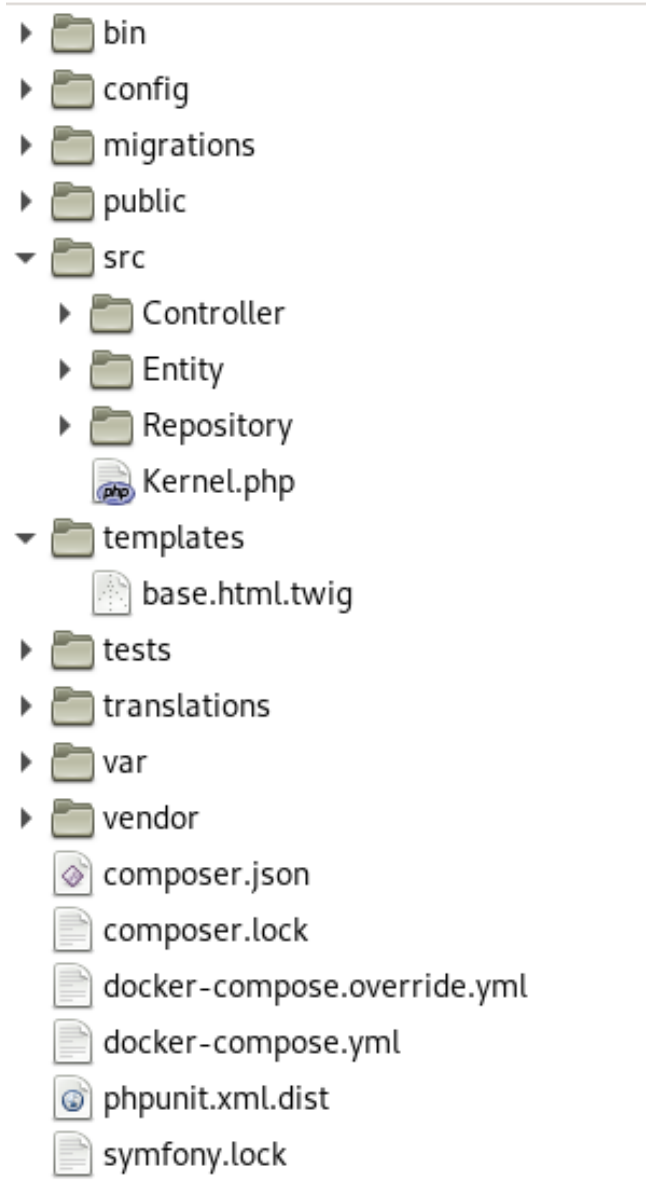
- `\l` : lister toutes les bases de données sur l'instance PostgreSQL
- `SELECT current_database();`
- `\dt` : afficher les tables de la base courante;
- `\h` : afficher la liste des commandes SQL

Rappels sur PostgreSQL

- Une instance de base de données PostgreSQL contient une ou plusieurs *bases de données*. Les *rôles* et quelques autres types d'objets sont partagés sur l'ensemble de l'instance ;
- Une connexion cliente au serveur ne peut accéder qu'aux données d'une seule base, celle indiquée dans la requête de connexion (sous réserve que le rôle possède le droit) ;
- **Une base de données contient un ou plusieurs *schémas*** nommés, qui eux contiennent des *tables*. Les schémas contiennent aussi des objets comme les types de données, fonctions, opérateurs, etc. Le même nom d'objet peut-être utilisé dans différents schémas sans conflit (rôle de *namespace* ou équivalent aux répertoires d'un système d'exploitation non imbriqués)

Pour plus d'informations, voir la documentation officielle de PostgreSQL

La structure d'un projet Symfony



Configuration de PHP

Via le profiler (env dev), accédez à la configuration de PHP `/_profiler/phpinfo`.

> L'environnement local de Symfony utilise directement php installé sur votre machine.
> Vous devez installer php (>8.1) et php-fpm (>8.1).

Règles et conventions d'organisation du projet

Standard PSR-4

Standard PSR-4: map entre nom complètement qualifié et *path* des fichiers PHP. **PascalCase** pour le nom des fichiers PHP ;

Standard PSR-1 et PSR-12 : Conventions de nommage

Casse

- Nom des classes: **PascalCase**. Ex: `User`
- Méthodes et attributs: **camelCase**. Ex: `nomDeMaMethode()` et `nomDeMonAttribut`
- Variables Twig, paramètres de configuration: **snake_case**. Ex: `http_status_code`
- Constantes: **SNAKE_CASE (All Caps)**. Ex: `NOM_DE_MA_CONSTANTE`

Suffixes, préfixes

- Interfaces suffixés par le mot-clef `Interface`. Ex: `UserLoginInterface`

Règles et conventions d'organisation du projet

Utiliser un outil !

On utilise PHP_CodeSniffer (`phpcs` et `phpbcs`), ou PHP CS (PHP Coding Standards Fixer) développé par l'équipe de Symfony, pour appliquer ces conventions, avec les coding standards Symfony !

Configuration d'une application Symfony

On peut configurer une application Symfony de 5 façons différentes :

1. **Attributs PHP**: depuis la version 8 de PHP. **À privilégier aujourd'hui !**

```
#[MonAttribut]
function foobar(){}
```

2. **Annotations** : utiliser les docblocks, **non natifs à PHP**. Basé sur les parsers/outils comme phpDocumentor, Doxygen. Introduits par l'ORM Doctrine.

```
/**
 * Ceci est un docblock
 * @MonAnnotation
 */
function foobar(){}
```

3. **YAML**;
4. **XML**;

Configuration d'une application Symfony

Mode	Avantages	Inconvénients
Attributs PHP	• Idem qu'Annotations • Natif ! • Accessible via l'API Reflection (code php) !	Plus aucun (2025) !
Annotations	• Rapides à écrire • Faciles à maintenir	• Pas natif à PHP • Juste des commentaires, ne *devraient* pas contenir de logique applicative
XML	• Strict, Validable • Structuré • autodescriptif	• Pas directement dans le code • Assez verbeux
YAML	Moins verbeux que XML	• Pas directement dans le code • Difficile à lire et à maintenir (indentation, structures complexes) • Sémantique pauvre
Script PHP	Performant	• Pas directement dans le code • Moins lisible • Plus difficile à maintenir

En choisir un et s'y tenir (cohérence).

Aujourd'hui, **privilégier les Attributs PHP** (ce qui est encouragé par le framework et la communauté)

Mettre en place les outils

- Analyseur statique de code PHPStan ;
- Linter (Formatage, code writing style et conventions) : PHP Code_Sniffer ou PHP CS, un linter configurable et extensible, développé par Fabien Potencier (Symfony). L'avantage c'est que l'outil contient et installe les coding standards de Symfony.

PHPStan

```
# Installer
composer require --dev phpstan/phpstan
# Analyser
vendor/bin/phpstan analyse -l 6 src
```

Installer PHP CS

A la racine du projet:

```
mkdir --parents tools/php-cs-fixer
composer require --working-dir=tools/php-cs-fixer friendsofphp/php-cs-fixer
```

Utiliser PHP CS

Inspecter et corriger uniquement les sources de notre application (dossier `src`)

```
#Checker sans appliquer les modifications
tools/php-cs-fixer/vendor/bin/php-cs-fixer check src
#Appliquer les corrections définies par les règles
tools/php-cs-fixer/vendor/bin/php-cs-fixer fix src
```

> A l'heure de publication de ce support, php-cs ne prend pas encore en charge la version 8.4 de PHP. On va tout de même l'utiliser avec le flag `PHP_CS_FIXER_IGNORE_ENV=1`. Soit :
> `PHP_CS_FIXER_IGNORE_ENV=1 tools/php-cs-fixer/vendor/bin/php-cs-fixer fix src`

Ressources

Symfony

- Symfony Docs: *what else ?*
- Installing & Setting up the Symfony Framework
- Official Symfony Book, version PDF payante, version HTML gratuite
- En route pour Symfony (fr), version française gratuite en ligne
- Liste des projets majeurs utilisant Symfony
- Cours complet sur Symfony 6

PostgreSql

- Connect To a PostgreSQL Database Server (Windows), comment se connecter à une db postgresql via la ligne de commande avec `psql` ou via pgAdmin (windows)
- pgAdmin, un client graphique postgresql
- pgAdmin, download, télécharger pgAdmin pour toutes les plateformes