

# **Présentation du framework Symfony**

# Table des matières

<b>I. Pourquoi un framework comme Symfony ?</b>	<b>3</b>
<b>II. Exercice : Quiz</b>	<b>7</b>
<b>III. La structure de Symfony</b>	<b>8</b>
<b>IV. Exercice : Quiz</b>	<b>10</b>
<b>V. Essentiel</b>	<b>11</b>
<b>VI. Auto-évaluation</b>	<b>11</b>
A. Exercice .....	11
B. Test .....	11
<b>Solutions des exercices</b>	<b>12</b>

# I. Pourquoi un framework comme Symfony ?

**Durée : 1 h**

Environnement de travail : Pc et connexion internet

## Contexte

Aujourd'hui il existe des centaines de langages en informatique, citons les plus connus, à commencer par Python qui en octobre 2022 est toujours le plus utilisé, mais aussi C, Java, C++, C#, Visual Basic, Javascript, PHP, SQL et Swift. Et dans cette jungle de langages où souvent l'on ne peut se contenter d'en maîtriser qu'un seul, les développeurs se spécialisent dans l'utilisation de bibliothèques ou de frameworks.

Si vous vous intéressez à ce cours c'est que vous avez déjà des notions de PHP ou alors que vous envisagez de l'apprendre. PHP est un langage omniprésent aujourd'hui dans le monde de l'IT. On estime que 79,2 % des sites internet utilisent ce langage. Il est vrai que les chiffres sont dopés par l'utilisation des CMS tels que WordPress qui sont entièrement écrits en PHP. Mais il n'empêche qu'apprendre ce langage est toujours un bon choix ! Même s'il a été décrié et ringardisé pendant un temps avec l'ascension entre autres du Javascript, PHP a su évoluer notamment avec sa dernière version majeure : PHP 8 ; et surtout grâce à ses dizaines de frameworks qui lui apportent chaque jour une plus grande facilité d'utilisation et une puissance redoutable ainsi qu'une sécurisation toujours de plus en plus essentielle pour une application internet. En 2017 il a été téléchargé plus d'un milliard de fois, il est donc l'un des plus utilisés au monde.

À travers ce cours nous découvrirons Symfony qui est le framework PHP le plus utilisé avec Laravel, dû à son ergonomie, son architecture MVC ainsi que le fait qu'il soit libre de droit. Symfony en est à sa version 6.2 qui est encore en développement en novembre 2022. Mais nous présenterons sa version 5.4 qui est la dernière version avant son évolution 6. Version qui est stable et reste un excellent choix pour créer une application web complexe ou encore une API (*Application Programming Interface*).

## Historique

C'est une société française qui a commencé en 2005 à développer Symfony : SensioLabs ! SensioLabs, spécialisé dans le développement web, s'est très vite rendu compte que les projets des clients ont souvent des fonctionnalités similaires. L'idée est donc née : créer un framework pour éviter de réécrire un même code ce qui prend un temps considérable. C'est en 2007 après une utilisation interne de ce framework nommé Symfony qu'ils ont décidé de l'ouvrir à la communauté PHP en le distribuant sous licence MIT, donc open source et libre.

Pour la petite histoire, SF qui est le logo de symfony sont les initiales de Sensio Framework, la branche qui l'a développé au sein de SensioLabs Symfony.

Le PHP, quant à lui, a été créé en 1994 par un développeur Groenlandais qui cherchait à analyser les connexions de son site web. Il n'imaginait certainement pas qu'aujourd'hui près de 300 millions de sites soient codés sous ce langage. PHP continue toujours d'évoluer, c'est aujourd'hui un langage moderne, il a su avec le temps se structurer et s'inspirer de Java mais aussi améliorer ses performances. PHP 7 a accru sa vitesse d'exécution de 50 %, la version 8 apporte la compilation à la volée et offre un gain de vitesse estimé à 45 % selon les applications.

## Qu'est-ce qu'un framework ?

Un framework, ou infrastructure logicielle en français, est un peu comme une boîte à outils. Les développeurs d'un framework partent d'un langage de programmation (pour Symfony le PHP) et développent des structures logicielles afin d'apporter un socle personnalisable mais qui reprend les fonctionnalités souvent utilisées afin d'éviter de les réécrire à chaque fois et ainsi de gagner du temps. C'est une bibliothèque de code qui constitue un squelette applicatif.

### Exemple

Si vous avez déjà programmé en PHP, vous avez certainement dû configurer un outil tel que le PDO (c'est un objet qui permet la connexion à la base de données en PHP), mais en installant Symfony, l'objet PDO a déjà été installé, vous n'avez donc pas besoin de créer sa classe et de la faire appeler. De simples paramètres dans le .env suffisent souvent à établir la connexion avec votre base de données. Ces paramètres tiennent en une ligne et sont déjà prérempli ! Ainsi un framework est là pour vous simplifier la vie et augmente à chaque version son squelette, cela permet d'optimiser les performances, la sécurité et vous fait gagner du temps puisque vous avez de moins en moins besoin de l'étoffer.

## Les avantages de Symfony

Comme on en a parlé en introduction, Symfony est écrit en PHP. Mais alors pourquoi l'utiliser plutôt que de rester en PHP, puisqu'il demande un certain apprentissage ?

Il est vrai que dans un premier temps, vous allez devoir vous former à l'utilisation de celui-ci. Mais très vite, vous rentabilisez le temps investi. Un des gros avantages c'est que la communauté Symfony est grande et qu'il vous sera facile de trouver du contenu ou des réponses aux problèmes que vous rencontrerez.

Voici quelques-uns des avantages qui pourront vous convaincre de l'utilité de celui-ci :

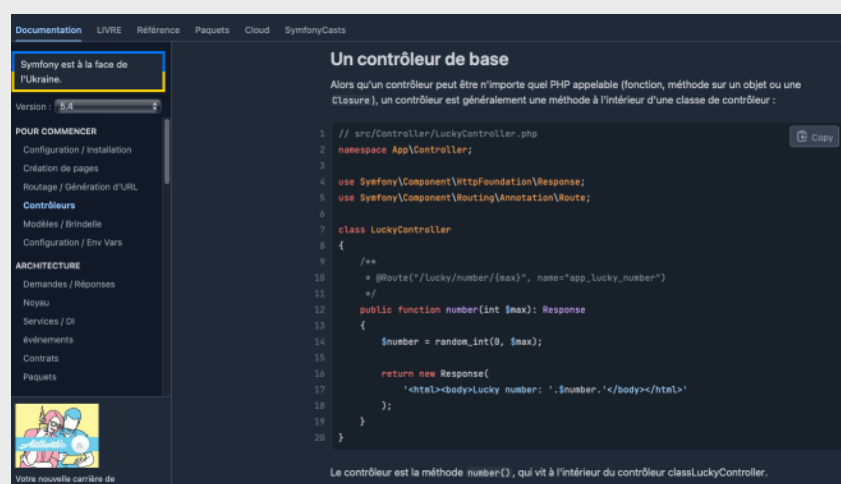
### La documentation :

Tout comme le framework, la documentation est également OpenSource et est disponible sur le site Github.com où l'ensemble des contributeurs peuvent modifier la documentation : GitHub<sup>1</sup>. C'est un avantage indéniable dans un monde où les technologies deviennent vite obsolètes et leurs documentations aussi. Cette documentation que vous retrouverez sur Symfony<sup>2</sup> est bien structurée. Elle possède des exemples clairs et est écrite de manière vivante. Chaque composant est détaillé et expliqué, c'est une vraie mine d'or. Bien qu'il y ait beaucoup de contenu en français elle est souvent par défaut en anglais mais si vous avez un navigateur moderne il vous convertira le contenu de la page en français, même s'il existe encore des petits bugs.

### Exemple

Allez sur contrôleur :

Une dizaine de points vous sont proposés ainsi que des sous-thèmes. Mais prenons, pour illustrer la simplicité de cette documentation, le premier point : « Un contrôleur de base ».



1 <https://github.com/symfony/symfony-docs>

2 <https://symfony.com/doc/5.4/templates.html>

**Copie d'écran d'un exemple de la documentation officielle de Symfony**

Source : capture d'écran du site : *Symfony*<sup>1</sup>

Vous constatez que vous avez un exemple concret et que vous pouvez même faire une copie avec le bouton en haut à droite. Ensuite les lignes 2, 4, 7, 12 et 16 sont décrites et expliquées.

Retranscrivez les explications !

Le contrôleur est la méthode `number()`, qui vit dans la classe du contrôleur `LuckyController`.

Ligne 2 : Symfony tire parti de la fonctionnalité d'espace de noms de PHP pour espacer toute la classe de contrôleur.

Ligne 4 : Symfony profite à nouveau de la fonctionnalité d'espace de noms de PHP. Via son propre chargeur de classe (autoloader) le mot-clé `use` va utiliser et importer en même temps la classe qui est requise ligne 16.

Ligne 7 : la classe peut techniquement s'appeler comme on veut, mais elle est suffixée avec `Controller` par convention.

Ligne 12 : la méthode `number` appelée par la route « `app_lucky_number` » récupère un paramètre directement depuis l'url, un entier nommé `$max`.

Ligne 16 : comme tout contrôleur, après le traitement d'une requête, la méthode doit créer et renvoyer un objet de type `Response`.

## Le serveur

Pendant tout votre développement vous n'avez pas besoin de déployer une architecture complexe pour communiquer avec un serveur web, vous pourrez utiliser le serveur de développement proposé par PHP que vous lancerez avec une simple ligne de commande.

## Les bibliothèques du framework Symfony

Symfony permet de créer des projets extrêmement extensibles, car ils ont une architecture modulaire et peuvent être agrémentés par les nombreux composants ou bundles que propose la riche communauté Symfony.

À l'installation, vous pouvez déjà choisir si vous voulez créer un projet comme un site web ou bien une API avec une ligne de commande différente. Car si vous avez besoin de construire une API, vous n'aurez pas l'utilité de certains composants comme Twig qui est un moteur de template (nous reviendrons dessus). Ainsi par défaut lorsque vous créez un squelette d'application web, Symfony vous installe des bundles et composants que vous retrouverez dans le fichier `composer.json`. Mais vous pouvez aussi à tout moment ajouter d'autres composants à l'aide de « `composer` ». Par exemple, bien que vous puissiez vous même créer un token, vous apprécierez sûrement d'installer « `lexik/jwt-authentication-bundle` », ce qui vous facilitera grandement la tâche.

## Composer

Est un gestionnaire de dépendances libres écrit en PHP et il est donc au cœur de la stratégie de Symfony. Il vous faudra donc installer Composer avant même de commencer votre projet, c'est un prérequis, tout comme PHP doit être installé avant. À ce propos, pour Symfony 5.4 il vous sera nécessaire d'avoir installé au moins la version 7.2.5 de PHP mais nous vous conseillons la version 8.1. Vous pourrez installer Composer en suivant les instructions sur `getcomposer`<sup>2</sup>. Une fois installé vous pourrez ajouter des centaines de dépendances. Il existe deux façon d'utiliser composer, soit en tapant en ligne de commande : `composer require « nom_de_la_dépendance »`, soit en ajoutant dans le fichier `composer.json` la dépendance et taper dans le terminal :

« `composer update` » qui va mettre à jour ou « `composer install` » pour installer votre fichier de configuration. Vous pouvez trouver toutes les dépendances libres sur le répertoire principal de Composer : Packagist<sup>3</sup>

1 <https://symfony.com/doc/5.4/controller.html#a-basic-controller>

2 <https://getcomposer.org/>

3 <https://packagist.org/>

## Twig

À moins que vous souhaitiez créer une API, Twig sera installé par défaut. Twig est un moteur de templates. Un moteur de template est une technique de programmation qui permet de tenir séparé l'interface graphique du reste du code, ainsi vous n'aurez pas de code PHP dans cette partie ou très peu, car Twig offre une syntaxe simple et particulière afin de pouvoir visualiser des éléments de la base de données, par exemple. C'est un vrai avantage car il permet d'utiliser le MVC dont nous reparlerons et dans un gros projet de permettre au développeur front-end de personnaliser les pages web sans avoir de véritables connaissances en PHP. Ainsi un template gèrera le code HTML, CSS, images, animations ainsi que le Javascript.

## Doctrine

Doctrine est l'ORM par défaut de Symfony. Qu'est-ce qu'un ORM (*Object-Relational Mapping*) ? Un ORM est un ensemble de classes permettant de manipuler les tables d'une base de données relationnelle, telle que MySQL ou PostgreSQL, comme s'il s'agissait d'objets. Ainsi vous n'aurez pas besoin de coder en SQL, même si connaître le SQL est un vrai plus, car vous pourrez être amené à coder du DQL dans un repository. Le DQL est un langage simplifié de doctrine pour communiquer avec la base de données. Même si nous parlerons plus tard de l'architecture de l'application, vous pouvez noter que Symfony organise à part vos entités qui représentent vos classes, c'est-à-dire vos objets qui seront utilisés dans la base de données. Ainsi, si vous passez par le terminal pour la création d'une entité, Symfony créera une entité et un repository associés à cette entité dans le dossier Repository. Ce repository vous permettra de faire appel à vos entités.

## Symfony Flex

Symfony Flex est apparu avec la version 4 de Symfony. C'est une nouvelle façon d'installer et de gérer les applications Symfony. Proprement dit, c'est un outil, un plugin Composer qui modifie le comportement des commandes require, update et remove. Ainsi il est possible de l'ajouter depuis Symfony 3.3, mais devient obligatoire depuis la version 4.

Comment cela fonctionne-t-il ? Lorsque vous exécutez « *composer require* » par exemple, l'application envoie une requête au serveur Symfony Flex avant d'essayer d'installer le paquet avec Composer. S'il n'y a aucune information sur ce paquet, l'installation se poursuit suivant la procédure habituelle de Composer. Mais s'il existe des informations spéciales sur ce paquet, le serveur Flex le renvoie dans un fichier appelé recipe (recette en français) et ajoute après l'installation des tâches automatisées.

Cela simplifie grandement la tâche : avant Symfony Flex, avec un « *composer require nom\_du\_bundle* » il était nécessaire d'ajouter une ligne au fichier AppKernel.php, de créer la configuration dans le fichier config.yml et d'importer le routing dans le fichier routing.yml.

### Complément La sécurité

La sécurité est un élément incontournable à notre époque, les failles sont diverses et nombreuses. En PHP il faut systématiquement penser à protéger chaque requête, formulaire ou donnée quelconque venant de l'utilisateur. Un oubli, une négligence, une coquille pourrait laisser passer des attaques CSRF, XSS ou injection SQL. Symfony propose justement un composant Sécurité basé sur 2 catégories : l'authentification et l'autorisation. Grâce à lui, Symfony intègre un pare-feu... Bien sûr, il faut toujours continuer à utiliser les bonnes pratiques sécuritaires mais les hackers concentrent leurs efforts sur d'autres cibles.

### Remarque Les performances

Les fonctions de cache HTTP natives permettent d'avoir de bonnes performances côté serveur et côté visiteur. Sans être trop technique, notez que l'APC ou OPcache permet d'éviter de recompiler le code à chaque appel et donc de fluidifier l'affichage des pages web.

## Facile à déboguer

Symfony propose plusieurs outils de débogage. Pour commencer, parlons du Profiler. Il s'agit d'une barre d'outils située en bas du site web en développement. En un clin d'œil vous avez des informations pertinentes comme l'état de la requête HTTP, la connexion ou s'il y a des messages d'exception. La page d'exception vous fournira également des messages précieux et vous affichera les lignes de codes qui posent un problème. Bien sûr, une bonne partie du temps du développeur sera toujours de chercher à déboguer, vous remplacerez sûrement vos échos par des dump mais symfony facilite vraiment cet aspect du travail.

## Respect des conventions

Les projets Symfony ont une architecture et une syntaxe uniques et rendent ainsi possible le lancement rapide d'un nouveau projet en limitant la phase d'apprentissage. Ainsi cela permet de se concentrer sur l'objectif métier. La maintenance est aussi nettement facilitée.

Symfony respecte les recommandations du PHP-FIG, ce qui permet à Symfony de s'interfacer avec les composants PHP. PHP ayant un important écosystème, cela représente un vrai plus. Par exemple, cela permet d'interroger des API, d'envoyer des e-mails ou d'autres éléments.

L'arborescence de la structure du projet offre une séparation claire entre le développement back-end et front-end. Tout en gardant une logique commune, les métiers peuvent se spécialiser.

Au-delà de toutes conventions intégrées à Symfony, la communauté du Framework a construit avec le temps un ensemble de bonnes pratiques qui permettent d'améliorer la qualité des projets.

## Exercice : Quiz

[solution n°1 p.13]

### Question 1

Un framework :

- ☐ Est une bibliothèque
- ☐ Est une bibliothèque plus grande
- ☐ N'est pas une bibliothèque mais inclut des bibliothèques

### Question 2

Symfony possède la plus grosse communauté de développeurs.

- ☐ Vrai
- ☐ Faux

### Question 3

Profiler est un outil de débogage.

- ☐ Vrai
- ☐ Faux

### Question 4

Symfony possède un outil utile en mode dev qui se nomme :

- ☐ Profiler
- ☐ Profil
- ☐ Thriller

### Question 5

Le PHP a été inventé par un développeur pakistanais.

- ☐ Vrai
- ☐ Faux

## III. La structure de Symfony

### La POO

Symfony utilise la POO (Programmation Orienté Objet), c'est-à-dire que tout est objet PHP. Pour relier les objets entre eux, PHP utilise autoload et namespace. Ainsi, pour utiliser une classe il faut que la page de celle-ci soit identifiée par son chemin d'accès et que la classe qui l'appelle possède le mot « *use* » suivi du chemin d'accès. Pour une entité, ce sera « *namespace App\Entity;* » et dans le contrôleur en haut sous son namespace l'entité sera appelée avec « *use App\Entity\Nom\_Entité* ».

Notez que App représente le dossier src\ qui fait partie de la structure de Symfony depuis Symfony 4, cette modification de nom est possible grâce à la PSR4 intégrée dans l'autoload du fichier composer.json

Notez avec l'exemple ci-dessous toutes les classes appelées grâce à l'auto-loader de Composer. Grâce à l'utilisation des namespaces et de « *use* », il n'est plus nécessaire d'utiliser require ou include.

```
1 <?php
2
3 namespace App\Controller;
4
5 use App\Entity\User;
6 use App\Form\UserType;
7 use App\Repository\UserRepository;
8 use Doctrine\ORM\EntityManagerInterface;
9 use Symfony\Component\HttpFoundation\Request;
10 use Symfony\Component\Routing\Annotation\Route;
11 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
12 use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
13
14 class UserController extends AbstractController
15 {
```

### Le MVC

Une fois que l'on a compris ce qu'est que la POO (sans rentrer dans les détails, car il faut tout un cours sur le sujet), il faut ensuite comprendre ce qu'est MVC (Modèle Vue Contrôleur). MVC est une architecture qui sert à découper l'application en modules avec trois grandes responsabilités.

- Les modèles gèrent l'accès aux données.
- Les vues sont les interfaces graphiques, ce que verra l'utilisateur.
- Les contrôleurs ont pour objectif de retourner une réponse.

**Le contrôleur**, comme son nom l'indique, contrôle tout mais n'a pas pour autant beaucoup de code, il va aller chercher les informations dont il a besoin pour retourner sa réponse.

Prenons un exemple :

Pour donner l'accès à un utilisateur, il va récupérer la requête, vérifier qu'elle est conforme au formulaire, l'envoyer à la base de données en passant par un service comme JWTtoken pour crypter le mot de passe, l'envoyer à la base de données, réceptionner la réponse pour accorder l'accès ou la refuser et enfin renvoyer la bonne page d'affichage.

**Les vues**, ce sont donc les pages qui ont été créées avec Twig et qui vont être dans un dossier templates.



**Le modèle** c'est la couche de données qui permettra d'y accéder via des repositories ou de modifier leurs structures via des entités. Dans notre exemple, le contrôleur ira justement l'interroger pour vérifier la conformité de notre formulaire.

## Arborescence de Symfony 5 et 6

Depuis Symfony 4, le framework a énormément évolué et a adopté une architecture plus moderne. Faisons le tour de cette structure.

- **Assets** : ce dossier contient tous les assets nécessaires pour le front, c'est-à-dire principalement les ressources graphiques : css, js, images, vidéos, etc.
- **Bin** : ce dossier contient les exécutables de votre projet ou de ses dépendances comme pour vider le cache Symfony ou mettre à jour la base de données. En tapant la commande **php bin/console** vous connaîtrez toutes les fonctionnalités proposées.
- **Config** : ce dossier contient toute la configuration des packages et routes. Vous les trouverez le plus souvent en YAML, mais ils peuvent aussi être en PHP ou en XML. C'est donc ici que l'on configurera la connexion à la base de données, que l'on personnalisera un service ou le système de sécurité.
- **Migrations** : dans ce dossier vous retrouverez les différentes migrations que Doctrine a générées dans la base de données.
- **Public** : ce dossier est le point d'entrée du site. Il contient le fichier `index.php` qui est le front contrôleur, c'est-à-dire le premier fichier appelé par le client. À côté on retrouve également les ressources compilées du dossier `assets`, des fichiers publics comme le `sitemap`, le `robots.txt` ou encore `l'htaccess`
- **Src** : ce dossier contient tout le code source PHP de votre projet. En plus de votre logique métier, on y retrouvera l'arborescence classique que propose Symfony :
  - **Controller** : c'est dans ce dossier que vous mettrez tous vos controllers qui sont les vrais moteurs de votre code et qui vont retourner les réponses au type voulu (Json, XML, etc.) suivant les méthodes choisies (GET, PUT, POST, DELETE, etc.) sur les routes désignées.
  - **Entity** : c'est dans ce dossier que vous mettrez vos entités qui correspondent à vos tables de la base de données.
  - **Repository** : c'est dans ce dossier que vous retrouverez chaque repository de chaque entité. Doctrine l'utilisera pour requêter la table de son entité.

Mais souvent vous y retrouverez d'autres dossiers suivant vos besoins, parlons des plus courants, `DataFixtures` et `Form`.

- **DataFixtures** : c'est dans ce dossier que vous mettrez vos fixtures. Les fixtures permettent d'insérer de fausses données en base de données pour faciliter un développement.
- **Form** : c'est dans ce dossier que vous mettrez vos formulaires. Symfony permet de relier les formulaires aux entités ce qui lui donne plus de puissance. On dit qu'ils sont « *mappés* » même s'il est possible de ne pas mapper un formulaire ou un champ.

Enfin vous retrouvez aussi le fichier `kernel.php` qui a migré dans ce dossier « *src* » depuis la version 5 de Symfony.

- **Kernel.php** (Kernel voulant dire noyau) : c'est lui qui va être appelé dès le début pour configurer l'ensemble de la stack Symfony (lire la configuration, tout instancier, appeler vos contrôleurs) afin de traiter la requête en réponse.

Comprenez bien que c'est dans ce dossier « *src* » que vous devez mettre tout votre code PHP, ainsi si vous devez créer des services par exemple, vous créerez un dossier de ce nom et y insérerez vos différents fichiers de service dedans.

- **Templates** : ce dossier contient tous vos templates qui par défaut sont en Twig et seront nommés `nom_de_fichier.html.twig`.

- Tests : ce dossier contient tous les tests unitaires ainsi que fonctionnels créés avec PHPUnit. L'arborescence devra être similaire à celle du dossier src/.
- Translations : ce dossier est facultatif, il est utilisé pour la traduction. Pour mettre en place ce système, il faudra installer le package translation.
- Var : ce dossier constamment réécrit et utilisé par le serveur sert principalement pour stocker le cache, les sessions ou encore les logs.
- Vendor : ce dossier est créé lorsque Composer installe les différentes dépendances pour le projet à partir de composer.json.
- composer.json : ce fichier contient tous les packages installés et mis à jour dans votre projet.
- composer.lock : ce fichier permet à Composer de retracer la version de chaque dépendance installée dans le projet.

Bien sûr, vous pourrez être amené à insérer d'autres dossiers ou fichiers dans votre structure comme des fichiers .env à la racine, ou docker si vous l'utilisez, mais vous avez ici une arborescence type de Symfony 5 ou 6. Cette structure plus ou moins imposée est un gros avantage car elle permet aux développeurs de s'y retrouver facilement dans n'importe quel projet et de gagner du temps dans la navigation entre les différents dossiers.

## Exercice : Quiz

[solution n°2 p.14]

### Question 1

Les templates sont en :

- ☐ Src
- ☐ Log
- ☐ Twig

### Question 2

Les namespaces sont une particularité de Symfony.

- ☐ Vrai
- ☐ Faux

### Question 3

Translations est un dossier :

- ☐ Obligatoire
- ☐ Facultatif
- ☐ Temporaire

### Question 4

Il n'est pas nécessaire de faire de tests avec Symfony.

- ☐ Vrai
- ☐ Faux

### Question 5

Lorsque vous créez un fichier PHP, vous devez le mettre dans un dossier du dossier src\.

- ☐ Vrai
- ☐ Faux

## V. Essentiel

À travers ce cours, vous avez pu faire connaissance du framework Symfony et découvrir quelques-uns de ses avantages. Vous vous posez peut-être la question « *est ce que cela vaut le coup que j'investisse du temps pour apprendre à maîtriser Symfony alors que je peux développer mes projets en PHP ?* »

C'est indéniable, une phase d'apprentissage sera nécessaire pour ce framework, vous pouvez certes construire un site en quelques minutes en suivant un tutoriel mais pour construire un site sérieux et l'adapter à vos besoins, c'est autre chose. Tout dépend donc de vos besoins.

Notez cependant que la communauté Symfony est très active, que la documentation est particulièrement bien faite et que vous pouvez sauter le pas. Un framework ne réinvente pas la roue mais vous permet de gagner du temps sur des tâches répétitives et parfois complexes, le concept est : une faible quantité de code pour de meilleures performances. Vous gagnerez du temps pour créer un CRUD (Create, Read, Update, Delete) par exemple, ou pour mettre en place un système de sécurité et bien d'autres choses encore. Mais surtout, vous gagnerez du temps sur la maintenance et l'amélioration du site et de ses fonctionnalités. Et puis si vous devez former quelqu'un par la suite, ou travailler avec d'autres personnes sur le projet, les choses pourraient vous être facilitées par les bonnes pratiques que Symfony vous apportera.

## VI. Auto-évaluation

### A. Exercice

Vous êtes un développeur back-end expérimenté dans une agence web de 6 personnes, donc une PME qui est en pleine croissance, où vous êtes amené à développer souvent en full stack.

#### Question 1

[solution n°3 p.15]

Pourquoi choisir le framework de Symfony serait une solution avantageuse pour vous et l'entreprise ?

#### Question 2

[solution n°4 p.15]

Quels sont également les avantages pour vos collègues qui sont développeurs front-end même s'ils ne maîtrisent pas le framework de Symfony ?

### B. Test

#### Exercice 1 : Quiz

[solution n°5 p.15]

##### Question 1

Symfony est sensible à l'injection SQL.

- ☐ Vrai
- ☐ Faux

##### Question 2

On peut supprimer le dossier vendor de l'arborescence d'un projet.

- ☐ Vrai
- ☐ Faux

##### Question 3

Quel dossier est constamment réécrit ?

- ☐ Var
- ☐ Vendor
- ☐ Config

Question 4

Un objet et une classe sont la même chose.

- ☐ Vrai
- ☐ Faux

Question 5


Où trouve-t-on les fonts dans Symfony ?

- ☐ Dans le dossier asset
- ☐ Dans le dossier src
- ☐ Dans le dossier template

## Solutions des exercices


**Exercice p. 7 Solution n°1****Question 1**

Un framework :

- ☐ Est une bibliothèque
- ☐ Est une bibliothèque plus grande
- ☒ N'est pas une bibliothèque mais inclut des bibliothèques
-  Un framework est une organisation, un squelette, une méthode de travail, tandis qu'une librairie n'offre que des fonctionnalités.


**Question 2**

Symfony possède la plus grosse communauté de développeurs.

- ☐ Vrai
- ☒ Faux
-  Symfony possède une communauté active et importante mais est loin d'être la plus grosse communauté de développeurs, il existe beaucoup d'autres langages informatiques.


**Question 3**

Profiler est un outil de débogage.

- ☒ Vrai
- ☐ Faux
-  Profiler est un outil de débogage dans Symfony.


**Question 4**

Symfony possède un outil utile en mode dev qui se nomme :

- ☒ Profiler
- ☐ Profil
- ☐ Thriller
-  Le profiler est un outil de développement puissant qui fournit des informations détaillées sur l'exécution de toute demande.

**Question 5**


Le PHP a été inventé par un développeur pakistanais.

- ☐ Vrai
- ☒ Faux
-  C'est un développeur groenlandais qui cherchait à analyser les connexions de son site et qui a donc créé le PHP.

## Exercice p. 10 Solution n°2


### Question 1

Les templates sont en :

- ☐ Src
- ☐ Log
- ☒ Twig
-  Les templates sont par défaut en Twig.


### Question 2

Les namespaces sont une particularité de Symfony.

- ☐ Vrai
- ☒ Faux
-  Les Namespaces, ou espaces de nom en français, sont une pratique courante en informatique : Java, C++, C# les utilisent aussi.


### Question 3

Translations est un dossier :

- ☐ Obligatoire
- ☒ Facultatif
- ☐ Temporaire
-  Translations est un dossier facultatif.


### Question 4

Il n'est pas nécessaire de faire de tests avec Symfony.

- ☐ Vrai
- ☒ Faux
-  Ce serait merveilleux mais l'importance des tests est de plus en plus soulignée à notre époque. Les tests sont très bien intégrés à Symfony via différentes dépendances et un binaire bin/phpunit.

### Question 5

Lorsque vous créez un fichier PHP, vous devez le mettre dans un dossier du dossier src\.

- ☒ Vrai
- ☐ Faux
-  Le dossier src\ doit contenir tout le code source PHP.

**p. 11 Solution n°3**

Votre connaissance en PHP est un vrai atout mais face aux nouveaux projets vous êtes obligés de réécrire des lignes de code qui vous prennent du temps et sont sources d'erreur, la partie sécuritaire vous accapare également beaucoup de temps. L'apprentissage d'un framework comme Symfony vous prendra un peu de temps au début, toutefois de façon raisonnable puisque vous maîtrisez la POO et PHP, mais vous permettra rapidement d'en gagner. Par exemple, mettre en place le PRG(Post\Redirect\Get) sur chaque page peut être très chronophage alors qu'il est important, particulièrement sur des pages de validation de données sensibles ou de paiement, or avec Symfony, le PRG est déjà intégré.

**p. 11 Solution n°4**


L'entreprise dans laquelle vous travaillez est petite et vous êtes souvent amené à donner un coup de main et à développer aussi en front-end, mais l'architecture de Symfony permet de séparer les métiers et vos collègues du front pourront, chaque fois que cela leur est possible, travailler sur les applications web. En effet, Twig, le moteur de templates par défaut de Symfony, ne demande que très peu de connaissances particulières, une fois mis en place. Ils pourront travailler sur le HTML, CSS, et Javascript qui sont leurs langages de prédilection sans difficultés. Il vous est également possible d'installer un préprocesseur comme Sass qui donne une puissance supplémentaire et permet une meilleure structuration au code CSS.

**Exercice p. 11 Solution n°5****Question 1**

Symfony est sensible à l'injection SQL.

☐ Vrai

☒ Faux


 Symfony apporte son lot de sécurisation, le composant form et l'utilisation de l'ORM Doctrine permet d'éviter ces injections.

**Question 2**

On peut supprimer le dossier vendor de l'arborescence d'un projet.

☐ Vrai

☒ Faux

 Si on supprime ce dossier, il faudra créer à nouveau un dossier vendor avec les dépendances re téléchargées.


**Question 3**

Quel dossier est constamment réécrit ?

☒ Var

☐ Vendor

☐ Config

 Le dossier var\ contient le dossier cache et le dossier log, qui sont constamment mis à jour.


**Question 4**

---

Un objet et une classe sont la même chose.

☐ Vrai

☒ Faux

 En POO, les objets sont des instances de classes.

**Question 5**

---

Où trouve-t-on les fonts dans Symfony ?

☒ Dans le dossier asset

☐ Dans le dossier src

☐ Dans le dossier template

 C'est dans le dossier asset que l'on retrouve tout ce qui est nécessaire pour le front.